Universidad Autónoma de Madrid
Escuela Politécnica superior

TRABAJO FIN DE GRADO

Desarrollo de una política de seguridad para el uso de aplicaciones Android en un contexto empresarial (SITI_5/1314)

Borja Simancas Ruesgas

Julio 2014

Desarrollo de una política de seguridad para el uso de aplicaciones Android en un contexto empresarial (SITI_5/1314)

Autor: **Borja Simancas Ruesgas**
Tutor: **David Arroyo Guardeño**
Ponente: **Gonzalo Martínez Muñoz**

Universidad Autónoma de Madrid
Escuela Politécnica Superior
Julio 2014

Resumen

La ciberseguridad se encuentra actualmente en las agendas de empresas y autoridades políticas internacionales y es considerada como una de las mayores preocupaciones mundiales. Los actores de los ciberataques, y sus motivaciones, han cambiado. Ahora priman las bandas de crimen organizado, los Estados que los utilizan como ciberarma, organizaciones privadas que buscan información secreta de empresas o colectivos guiados por principios ideológicos o políticos. También ha crecido el tráfico de datos personales que violan la privacidad de los ciudadanos. Se trata de riesgos reales que causan y/o podrían causar devastadores efectos, puesto que los ciberataques podrían hacer fallar los servicios públicos esenciales, provistos por las infraestructuras críticas de un país o, incluso, poner en riesgo la propia seguridad nacional.

Durante el último lustro, muchos de los países de nuestro entorno han reconocido la importancia estratégica de disponer de un ciberespacio seguro para garantizar la prosperidad económica, social y cultural. La Estrategia Nacional de Ciberseguridad debe transmitir dicha importancia, pero además, debe servir de guía para los responsables de la ciberseguridad nacional y servir como instrumento de disuasión para sus potenciales transgresores. Actualmente en España no existe, pero en los últimos meses se han desarrollado proyectos de una índole similar por parte del Departamento de Seguridad Nacional de España.

A su vez, en los últimos años ha existido una gran emergencia por parte de los dispositivos móviles y de los sistemas distribuidos. El caso límite de este ejemplo es la nube. La nube implica una falta de control sobre los datos, puesto que están alojados en lugares ajenos al propietario. Igual de importante es la transmisión de esos datos a través de conexiones de Internet, con lo que se hacen necesarios nuevos sistemas de cifrado más seguros y eficientes. Si a este factor se le añade el dilema "Bring Your Own Device" (BYOD) , el cual refleja la tendencia actual de usar el mismo dispositivo en el ámbito laboral y privado, se incrementa la posibilidad de entrañar erosión en los niveles de seguridad y protección de documentos confidenciales.

Existe una gran preocupación a nivel empresarial debido a la cantidad de información confidencial que se gestiona con dispositivos móviles. El presente documento pretende mostrar el conjunto de herramientas existentes que dispone un administrador de seguridad para proteger los activos de información en un contexto BYOD, así como su uso. En primer lugar se ha definido en que consiste una política de seguridad. A continuación, se ha entrado en detalle en el sistema operativo fundamental de este documento, Android. Posteriormente se han presentado las herramientas con las que configurar un dispositivo Android en un entorno empresarial. Por último, se ha realizado un estudio de los sistemas de detección de intrusos IDS (Intrusion Detection Systems) y de los Mobile Device Management (MDM).

Palabras clave: MDM, Android, *firewall*, IDS, *smartphone*, SSH, VPN, BYOD, ciberseguridad, Linux, *rooteo*.

Abstract

Cybersecurity is currently on the main concerns of international companies and political authorities and it is considered one of the most relevant challenges. Actors from cyber attacks, and their motivations have changed. The current scenario of cyberthreats comprises organized criminal gangs, states that use it as cyberweapons, private organizations trying to get secret information from companies, and ideological and political groups. It has also grown the traffic of personal data that could imply a violation of the privacy of citizens. These are real risks that cause and/or could cause devastating effects. In fact, cyberattacks could cause failure of essential public services, as critical infrastructures of countries or they could even jeopardize national security itself.

During the last five years many of the countries in our geopolitical environment have recognized the strategic importance of a secure cyberspace to ensure economic, social and cultural prosperity of the different states. The National Cyber Security Strategy should convey this strategic importance, but also should provide guidance to those responsible for the control and management of the national cybersecurity and serve as a deterrent to potential transgressors. Currently in Spain there is not a National Cybersecurity Strategy. However, in recent months they have developed projects of a similar nature by the Spain Homeland Security department.

Meanwhile, in recent years there has been a great emergence of mobile devices and distributed systems. An extreme case of this last technology is the cloud. The cloud implies a lack of control over the data, since they are staying in places outside the owner. Equally important is the transmission of the data through Internet connections, so new systems more secure and efficient encryption are necessary. If we add to this factor the irruption of the so-called "Bring Your Own Device" (BYOD) dilemma, there exists an increasing level of risk affecting the integrity of security means and the protection of confidential information.

In the enterprise context the generation and management of information by means of mobile devices is increasing day by day. This fact represents an advantage from the perspective of technology usability, but a risk from the point of view of guaranteeing the fulfillment of the information security goals . This work show in a comprehensive way the main risk associated to the BYOD environment, and it summarizes the set of tools available to a security administrator of a company to implement an adequate security policy for using mobile devices. This work is focused on the most popular OS for mobile devices, Android. In this regard, we discuss a meaningful set of o properly configure an Android device in an enterprise environment. Finally, and in addition, we have studied how to enforce our security policy through Intrusion Detection Systems (IDS) and an adequate Mobile Device Management (MDM) system.

Key words: MDM, Android, firewall, IDS, smartphone, SSH, VPN, BYOD, cibersecurity, Linux, rooted.

Agradecimientos

Me gustaría expresar mi agradecimiento a todas aquellas personas que con su ayuda han colaborado y ayudado en la realización del presente trabajo.

Mi mayor agradecimiento es hacia mi tutor, David. Sin él no habría sido posible llevar a cabo este proyecto. Sus conocimientos, su apoyo y su implicación, de una forma totalmente desinteresada, han sido la razón por la que este proyecto ha alcanzado la meta.

A todos mis compañeros de la carrera, en especial a Carlos, Iván y Alejandro. Porque han sido cuatro años juntos y seguro que sin el apoyo mútuo no habríamos llegado hasta aquí.

Por último, y no por ello menos importante, me gustaría agradecer a toda mi familia el apoyo que me han brindado a lo largo de la carrera. Ellos son los que han tenido que aguantar día tras día apoyándome, tanto en los momentos buenos como en los momentos malos, de una manera estoica.

A todas las demás personas, que de una manera u otra han formado parte de mi vida a lo largo de esta carrera, gracias.

Índice general

Índice general **9**

Índice de figuras **13**

1. Introducción **15**

2. Definición de una política de seguridad **19**

2.1. Conceptos relativos a la gestión 20

2.2. Clasificación de la información 20

2.3. Gestión del riesgo 21

2.4. Normativas y estándares 23

2.4.1. Principios de seguridad del NIST 23

2.4.2. Estándar ISO27001 24

3. Android **27**

3.1. ¿Qué es Android? 27

3.2. Arquitectura de Android 29

3.3. Principales amenazas y mecanismos de seguridad en Android 31

3.3.1. Amenazas en Android 31

3.3.2. Mecanismos de seguridad 33

3.4. ADB 36

3.5. Android en entornos empresariales 37

4. Implementación de la política de seguridad **45**

4.1. Obtención de permisos de superusuario: rooteo de un dispotivo Android . . 45
4.2. Configuración de las interfaces de comunicación 46
4.2.1. SSH 46
4.2.2. VPN 47
4.2.3. Aplicar políticas de firewall 52
4.2.4. Deshabilitar Bluetooth 57
4.3. Configuración de sistema en Android 58
4.3.1. Aplicar una política de password acorde a la compañía 58
4.3.2. Deshabilitar la instalación de software 59
4.3.3. Deshabilitar tarjeta SD 60
4.3.4. Deshabilitar binarios innecesarios 60
4.3.5. Eliminar software innecesario 61
4.3.6. Remote Wipe 61
4.3.7. Antivirus 62

5. Monitor de tráfico **63**
5.1. Sistema de detección de intrusos (IDS) 63
5.2. IDS existentes en el mercado 65
5.3. Instalación y configuración de Snort 67
5.3.1. Capturando el tráfico 68
5.3.2. Creación de reglas propias y testeo de ellas 68

6. Sistemas MDM **73**
6.1. MDM existentes en el mercado 74

7. Líneas futuras **79**

8. Conclusiones **81**

Bibliografía **83**

Glosario **85**

Índice alfabético **86**

A. Cláusulas Normativa ISO 27001 **87**

B. Certificados digitales en Android **89**

Índice de figuras

2.1. Clasificación de la información 21
2.2. Análisis de riesgo 22
2.3. Comparativa de los análisis de riesgo 22
2.4. Ejemplo de matriz de riesgos 23
2.5. Esquema PDCA 24

3.1. Evolución Android 27
3.2. Cuota de mercado de Android 28
3.3. Arquitectura de Android 29
3.4. Arquitectura de aplicación en Android 30
3.5. Procesos de dos aplicaciones sin compartir recursos 33
3.6. Procesos de dos aplicaciones compartiendo recursos 33
3.7. Flujo de los permisos en una apk 40
3.8. Ejemplo de ComDroid 43
3.9. Ejemplo de acceso a la cámara con permisos 43
3.10. Ejemplo de acceso a la cámara sin permisos 44

4.1. Esquema SSH 47
4.2. Esquema VPN 48
4.3. Diagrama del esquema VPN 50
4.4. Configuración del servidor VPN 51
4.5. Configuración de los usuarios VPN 51

4.6. Esquema genérico de firewall 52

4.7. Flujo de datos de IPTables 54

4.8. Flujo de datos de IPChains 55

4.9. Device Manager de Google 62

4.10. AVG 62

5.1. Sistema de detección de intrusos 63

6.1. Arquitectura de un MDM 73

6.2. Ejemplo de ownMDM 77

B.1. Intercambio de información con el certificado digital 89

Capítulo 1

Introducción

El desarrollo de los dispositivos y comunicaciones móviles y de las tecnologías inalámbricas en los últimos años ha revolucionado la forma de trabajar y comunicarse. El uso creciente de estas tecnologías sitúa a los dispositivos móviles como uno de los objetivos principales de las ciberamenazas.

Uno de los campos de desarrollo software de mayor crecimiento en los últimos años es, sin duda, el de las aplicaciones para dispositivos móviles en general y, específicamente, de dispositivos con sistema operativo Android. Dicho desarrollo, tal y como ocurrió (y sigue ocurriendo) con el desarrollo web, ha ido acompañado de un aumento más que considerable de ataques sobre las aplicaciones desarrolladas. Dichos ataques están fundamentados en problemas de seguridad derivados de prácticas de programación y de configuración de aplicaciones no adecuadas.

En un sistema operativo clásico, las aplicaciones son ejecutadas con distintos niveles de seguridad; los procesos más importantes y vitales del sistema corren siempre en un entorno seguro y muy poco accesible, mientras que la mayoría de los procesos de usuarios, es decir, el software usado a diario, son ejecutados en un entorno en donde están mucho más expuestos. Este tipo de arquitectura es necesaria ya que de otra manera, almacenar un simple documento de texto en el disco sería una tarea muy tediosa en cuanto a los permisos se refiere. Por ejemplo, UNIX se basa en el sistema ”setuid” y ”seguid”, los cuales son permisos de accesos que se asignan a ficheros o directorios de forma temporal. Con respecto a Windows, los permisos se gestionan en base a la creación y gestión de perfiles (Active Directory). En ambos casos se sigue el precepto del principio de mínimo privilegio.

Precisamente las capas de software más expuestas son en las que los desarrolladores de virus y *malware* suelen enfocarse, debido a que son mucho más vulnerables. Usando esta particularidad, los virus pueden comenzar a extenderse y posteriormente atacar las capas del sistema más seguras, donde residen los procesos vitales del sistema, afectando al rendimiento de nuestro ordenador, y lo que es más importante y peligroso: robarnos la información. Este tipo de ataque es conocido como Escalada de privilegiosescalada de privilegios.

Por otra parte, el método de infección usado en *smartphones* y *tablets* es bastante diferente. En este sentido, Android ha sido diseñado con dos capas de seguridad: la primera de ellas totalmente restringida al usuario donde se ejecutan y almacenan todos los procesos y archivos vitales para el sistema; y la segunda, donde reside la información personal del

usuario y las aplicaciones y configuraciones que ha descargado e instalado. De este modo, cuando un *smartphone* es infectado con virus o *malware*, las acciones de estos sólo tienen efecto sobre los datos y configuraciones del usuario. Es por ello que la implementación de técnicas como el *rooteo* en dispositivos Android siempre es una tarea muy peligrosa, ya que deja una puerta abierta a la entrada de virus a los niveles más seguros del sistema.

La proliferación de dispositivos móviles en los últimos años, junto al aumento de las capacidades, prestaciones y posibilidades de utilización de los mismos, hace necesario evaluar en profundidad la seguridad ofrecida por este tipo de dispositivos, así como de los mecanismos de protección de la información que gestionan, dentro de los entornos de Tecnologías de la Información y las Comunicaciones (TIC). Se considera dispositivo móvil aquél dispositivo de uso personal o profesional de reducido tamaño que permite la gestión de información y el acceso a redes de comunicaciones y servicios, y que habitualmente dispone de capacidades de telefonía, tanto de voz como de datos. La utilización y amplia adopción, sin precedentes, de los dispositivos móviles como herramientas básicas de productividad en el ámbito profesional, junto a su utilización simultánea en el ámbito personal, hacen necesario que las organizaciones realicen una gestión minuciosa, exhaustiva y continua de los mismos, acorde con las políticas de seguridad de la organización.

Esta estrategia, conocida como movilidad empresarial, permite a los usuarios y empleados de las organizaciones llevar a cabo sus actividades diarias de negocio a través de dispositivos móviles que aprovechan las tecnologías que facilitan el acceso remoto a los datos corporativos, incrementando su eficiencia y productividad con independencia de la ubicación física en la que se encuentran y con mayor flexibilidad en los desplazamientos.

Las soluciones tecnológicas que permiten la gestión de los dispositivos móviles a nivel empresarial se conocen como MDM, de sus siglas en inglés, Mobile Device Management. Estas soluciones permite gestionar de forma eficiente la diversidad y el despliegue masivo, dinámico y a gran escala de dispositivos móviles en una organización, con un enfoque principalmente orientado a incrementar su seguridad, y mejorando colateralmente la productividad del usuario final.

En este documento se analizarán las principales amenazas que pueden surgir a la hora de desplegar un conjunto de aplicaciones Android como base del trabajo colaborativo en el seno de una empresa. A tal efecto se estudiará en profundidad el sistema de gestión de control de acceso y distribución de privilegios en el entorno Android, al mismo tiempo que se implantarán (a través de los protocolos y primitivas criptográficas pertinentes) las medidas precisas para establecer canales de comunicación seguros, implementar una política conveniente de contraseñas para acceso a terminales y/o aplicaciones, limitar el acceso a la administración de dispositivos, habilitar control remoto para funcionalidades de seguridad, etc.

Por otro lado, el presente documento realiza un análisis detallado de las características, funcionalidades y mecanismos de seguridad existentes en las diferentes soluciones MDM (compuestas de productos y servicios) disponibles en la industria actualmente; así como un análisis detallado de los mecanismos y la configuración de seguridad recomendados para uno de los principales sistemas operativos de dispositivos móviles utilizados en la actualidad, Android, con el objetivo de reducir su superficie de exposición frente a ataques de seguridad.

En el presente documento se va a definir qué es una política de seguridad. Posteriormente, se realizará un análisis del sistema operativo Android. A continuación, se desarrollará una política de seguridad en base a la definición realizada anteriormente y a las

necesidades que tenga Android. Y, por último, para acabar con el desarrollo de la política de seguridad se estudiarán herramientas de monitorización de sistemas y de administración de dispositivos.

Capítulo 2

Definición de una política de seguridad

En cualquier empresa la información es un activo fundamental para la prestación de sus servicios y la toma de decisiones eficientes, razón por la cual debe existir un compromiso expreso de protección de sus propiedades más significativas como parte de una estrategia orientada a la continuidad del negocio, la administración de riesgos y la consolidación de una cultura de seguridad [1].

En función de las necesidades actuales de cada empresa, se debe implementar un modelo de gestión de seguridad de la información como una herramienta que permita identificar y minimizar los riesgos a los cuales se expone la información, ayudar a la reducción de gastos operativos y financieros, establecer una cultura de seguridad y garantizar el cumplimiento de los requerimientos legales, contractuales, regulatorios y de negocio vigentes.

El proceso de análisis de riesgos de los activos de información es el soporte para el desarrollo de las Políticas de Seguridad de la Información y de los controles y objetivos de control seleccionados para obtener los niveles de protección esperados.

La política establecida debe ser revisada con regularidad como parte del proceso de revisión gerencial, o cuando se identifiquen cambios en el negocio, su estructura, sus objetivos o alguna condición que afecten la política, para asegurar que sigue siendo adecuada y ajustada a los requerimientos identificados.

A la hora de definir una política de seguridad en un sistema general de información se deben seguir los siguientes pasos:

- Identificación de los activos del sistema, en concreto, los activos de información.
- Desarrollo e implementación de políticas, estándares, directrices y procedimientos.
- Clasificación de la información.
- Gestión de riesgos.
- Identificación de amenazas respecto a la confidencialidad, integridad y disponibilidad de la información.
- Clasificación de los activos y ponderación de sus vulnerabilidad.

A continuación se procederá a explicar sucintamente los pasos anteriores. Para ello se definirán los conceptos fundamentales en el ámbito de la seguridad y se analizarán los pasos más determinantes de los anteriormente enumerados.

2.1. Conceptos relativos a la gestión

El ciclo de vida de un sistema de seguridad engloba las siguientes etapas:

1. Fase de iniciación.
2. Fase de desarrollo/adquisición.
3. Fase de implementación.
4. Fase de operación/mantenimiento.
5. Fase de terminación.

Por otro lado, existen tres grandes características en los sistemas de información los cuales son los que más preocupan a la hora de la definición de una política de seguridad:

- **Confidencialidad**. Consiste en evitar el acceso no autorizado a la información.
- **Integridad**. Su objetivo es otorgar consistencia interna y externa de la información.
- **Disponibilidad**. Su principal fin es el conseguir que el sistema esté operativo siempre que se le precise.

La combinación de estos tres elementos permite a la empresa conseguir de forma eficiente sus objetivos y transmite al cliente la sensación de sistema confiable.

A la hora de la definición de la política de seguridad, es muy importante que el responsable de seguridad tenga claro los aspectos que la van a determinar. Para ello debe plantearse preguntas como: si es posible perder clientes porque una mala política impida una correcta funcionalidad, si existen antecedentes de pérdidas de disponibilidad o monetarias por incidentes de seguridad, conocer los potenciales enemigos del sistema, si existe información confidencial accesible desde Internet, si es importante establecer una estructura de capas de seguridad en la organización, etc.

Por último, es deseable que el sistema posea las características de autenticación, auditabilidad y autorización (las tres reglas de oro -Au-). Si dichas características se cumplen se puede asegurar que los usuarios son quienes dicen ser, que exista la capacidad de supervisar las tareas efectuadas por los usuarios y que se pueda autorizar a los distintos usuarios a ciertas tareas.

2.2. Clasificación de la información

En toda empresa es muy importante la definición de estrategias como los Planes de Continuidad de Negocio (BCP) y los Planes de Recuperación ante Desastres (DRP).

Los BCP son planes logísticos para la práctica de cómo una organización debe recuperar y restaurar sus funciones críticas parcial o totalmente interrumpidas dentro de un tiempo predeterminado después de una interrupción no deseada o desastre.

Por otro lado, los DRP son procesos de recuperación que cubren los datos, el hardware y el software crítico, para que un negocio pueda comenzar de nuevo sus operaciones en caso de un desastre natural o causado por humanos.

La clasificación de la información es crucial a la hora de diseñar BCP y DRP. Una buena clasificación es una prueba del compromiso respecto a la seguridad por parte de la empresa. A su vez, ayuda a localizar información crítica y permite inferir los métodos de protección a aplicar a cada tipo de información. Por otro lado, es un elemento legal y regulador.

Llevando esta clasificación a la práctica, es el dueño de la información el encargado de valorar el grado de importancia de la misma. De forma global, se puede clasificar la información como información gubernamental o como información del sector privado. Dentro de la información gubernamental, en función del peligro que entraña el carácter público de la información ésta puede ser: información no clasificada, información sensible pero no clasificada, información confidencial, información secreta e información de alto riesgo.

No obstante, una organización establece las distintas categorías en función de sus exigencias respecto a la confidencialidad, integridad y disponibilidad (ver Fig.2.1).

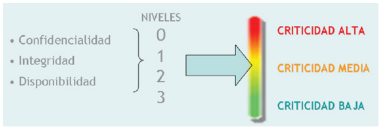

Figura 2.1: Clasificación de la información

El procedimiento que se debe seguir en la clasificación de la información es el siguiente:

1. Identificar al administrador de seguridad.
2. Especificar los criterios para clasificar y etiquetar la información.
3. Clasificación efectuada por el propietario de la información. Posteriormente debe ser evaluada por el supervisor.
4. Especificar y documentar cualquier excepción.
5. Especificar los controles a aplicar a cada uno de los niveles.
6. Concretar los procedimientos para desclasificar la información o para transferirla.
7. Crear un programa de concienciación respecto a la necesidad de clasificar la información.

2.3. Gestión del riesgo

El objetivo de gestionar el riesgo es reducirlo a un nivel aceptable para la organización. Es importante tener claro que el riesgo nunca desaparece. Se trata de analizar que nivel de riesgo puede tolerar la empresa de modo que siga siendo operativa.

El análisis de riesgo se realiza para cuantificar el impacto de amenazas potenciales, con lo que se pone un precio a la pérdida de la funcionalidad (ver Fig.2.2). A su vez, la identificación del riesgo y la valoración del mismo permiten confeccionar una táctica de mitigación del riesgo. Todo esto influye a la hora de seleccionar nuevos equipos informáticos, así como a la asignación de recursos.

Figura 2.2: Análisis de riesgo

Existen dos tipos de análisis de riesgo:

- **Análisis de riesgo cuantitativo**. Consiste en la asignación de valores monetarios a los distintos activos y a la pérdida derivada de la concreción de una amenaza. Este análisis exige la existencia de un jefe y distribución de tareas. Cabe destacar que no es posible efectuar un análisis de riesgo cuantitativo de forma plena.

- **Análisis de riesgo cualitativo**. Este análisis está más orientado al escenario de trabajo. Obtiene un cálculo más precio y completo. Este análisis está basado en la determinación de frecuencias e impacto de amenazas.

Propiedad	Análisisis Cuantitativo	Análisis Cualitativo
Análisis de coste/beneficio	Sí	No
Alto coste económico	Sí	No
Puede automatizarse	Sí	No
Grado de intuición	Bajo	Alto
Cálculos complejos	Sí	No
Volumen de información precisado	Alto	Bajo
Tiempo/trabajo precisado	Alto	Bajo
Fácil comunicación	Alto	Bajo

Figura 2.3: Comparativa de los análisis de riesgo

En el seno empresarial es de suma importancia determinar el impacto sobre la organización, así como la probabilidad con la que un ataque puede producirse y la probabilidad de que el ataque se traduzca en una pérdida (ver Fig.2.4).

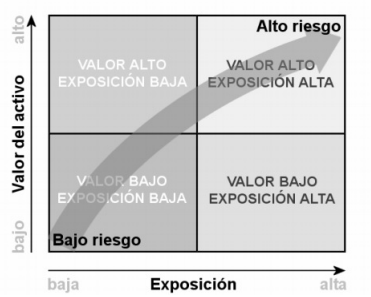

Figura 2.4: Ejemplo de matriz de riesgos

Para ello se deben realizar controles de riesgo. Existen varios tipos de controles:

1. **Controles disuasorios**. Estos controles reducen la probabilidad de que acontezca un ataque.
2. **Controles preventivos**. Protegen las vulnerabilidades del sistema de modo que un ataque presente la mínima repercusión. A su vez, inhiben tentativas contra la política de seguridad.
3. **Controles correctivos**. Reducen el impacto de un ataque.
4. **Controles detectores**. Desvelan ataques y disparan ataques preventivos o correctores, según el caso. También informan respecto a violaciones o intentos de violación de la política de seguridad.

2.4. Normativas y estándares

Las organizaciones necesitan demostrar que realizan una gestión efectiva y competente de la seguridad de los recursos y datos que gestionan en multitud de ocasiones. Para poder demostrar esto, existen las normativas y los estándares.

Una de las entidades más reconocidas a nivel mundial en el ámbito de los estándares de la tecnología es el Instituto Nacional de Estándares y Tecnologías (NIST).

2.4.1. Principios de seguridad del NIST

El Instituto Nacional para estándares y tecnología, publicó en Junio de 2001, 33 principios de seguridad que deben ser tenidos en cuenta en el diseño y se deben cuidar durante el ciclo de vida del sistema de seguridad. Estos principios fueron revisados 2004: [2]. Alguno de estos principios son:

- **Principio 1**. Establecer una política sólida como base del diseño del sistema.
- **Principio 2**. Tratar la seguridad como parte constituyente del sistema en global.
- **Principio 6**. Asumir todo sistema externo como inseguro.
- **Principio 7**. Identificar posibles compromisos entre reducir riesgos e incrementar costes, así como empeorar el rendimiento en otras actividades.

- **Principio 8**. Implementar una seguridad por capas.
- **Principio 20**. Aislar puntos de acceso al exterior frente a elementos críticos.
- **Principio 21**. Separar claramente los dispositivos del resto de equipos.
- **Principio 25**. Minimizar el número de elementos confiables.
- **Principio 32**. Autenticar usuarios y procesos con el objeto de asegurar decisiones apropiadas de control de acceso.
- **Principio 33**. Emplear identidades únicas con el objetivo de posibilitar la auditabilidad.

2.4.2. Estándar ISO27001

Las normas ISO 27000 centran el foco en la gestión de la seguridad. Dichas normas son una familia de estándares de ISO que proporcionan un marco para la gestión. Una de las normas más importante de esta familia es la ISO 27001: Implementación del SGSI/Certificación (Procesos de auditoría para la Implementación).

La ISO 27001 es un modelo para establecer, implementar, poner en marcha, supervisar, revisar, mantener y mejorar un Sistema General de Seguridad de la Información (SGSI).

Dicha norma está basada en un modelo PDCA: "PlanDoCheckAct" (ver Fig.2.5). Los cuatro pasos del modelo se pueden desglosar de la siguiente forma:

- **Plan: establecer el SGSI**. Se debe establecer la política de seguridad, así como los objetivos, procesos y procedimientos adecuados.
- **Hacer: implementar y poner en marcha el SGSI**. Implementar y desarrollar las políticas del SGSI, los controles, los procesos y procedimientos.
- **Verificar: evaluar**. Y, donde sea aplicable, medir el grado de satisfacción con respecto a la política, objetivos y experiencia del SGSI. A su vez, se deben documentar los resultados y enviar a Dirección para la revisión.
- **Actuar: mantener y mejorar el SGSI**. Emprender acciones correctivas y preventivas en base a auditorías internas y revisiones de la gestión, con el objeto de lograr una mejora continua del SGSI.

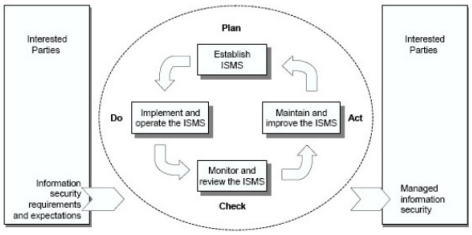

Figura 2.5: Esquema PDCA

Los requerimientos contenidos en esta norma son aplicables a todo tipo de empresa, indistintamente de su naturaleza o tipo.

A la hora de aplicar la norma, no es aceptable que se excluyan ninguno de los requerimientos señalados en las cláusulas 4-8 (VER ANEXO). En caso de la exclusión de algún control, será necesario demostrar que las exclusiones no menoscaban la capacidad de la organización y/o su responsabilidad/seguridad de la información en concordancia con las exigencias derivadas del análisis de riesgos y las medidas regulatorias aplicables.

Capítulo 3

Android

3.1. ¿Qué es Android?

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. Inicialmente fue desarrollado por Android, Inc. Google respaldó el desarrollo económicamente y posteriormente adquirió la empresa en 2005. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

Actualmente, existe una gran comunidad de desarrolladores codificando aplicaciones para extender la funcionalidad de los dispositivos. A fecha de hoy, se ha llegado al millón de aplicaciones disponibles para la tienda de aplicaciones oficial de Android: Google Play.

Figura 3.1: Evolución Android

En cuanto a la cuota de mercado, según el informe acerca del comportamiento mundial de *smartphones* en 2013 de la consultora IDC se confirma que Android lidera con una cuota de mercado de aproximadamente 80 %, lo que representa un incremento del 9,6 % frente a 2012. (ver Fig.3.2)

Top Five Smartphone Operating Systems, Shipments, and Market Share, 2013 (Units in Millions)

Operating System	2013 Shipment Volumes	2013 Market Share	2012 Shipment Volumes	2012 Market Share	Year-Over-Year Change
Android	793.6	78.6%	500.1	69.0%	58.7%
iOS	153.4	15.2%	135.9	18.7%	12.9%
Windows Phone	33.4	3.3%	17.5	2.4%	90.9%
BlackBerry	19.2	1.9%	32.5	4.5%	-40.9%
Others	10.0	1.0%	39.3	5.4%	-74.6%
Total	**1009.6**	**100.0%**	**725.3**	**100.0%**	**39.2%**

Figura 3.2: Cuota de mercado de Android

Las principales características de Android son:

- **Diseño de dispositivo**. La plataforma de Android es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL y diseño de teléfonos tradicionales.
- **Almacenamiento**. SQLite, una base de datos sencilla que es usada para propósitos de almacenamiento de datos.
- **Conectividad**. Android soporta gran variedad de tecnologías de conectividad.
- **Mensajería**. SMS y MMS son formas de mensajería, incluyendo la Android Cloud to Device Messaging *framework* (C2DM).
- **Navegador web**. El navegador web incluido está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome.
- **Soporte de Java**. En Android no existe una máquina vitual Java. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la máquina virtual Dalvik.
- **Soporte multimedia**. Android soporta gran variedad de formatos multimedia.
- **Soporte para hardware adicional**. Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro y aceleración por GPU.
- **Entorno de desarrollo**. Android incluye un emulador de dispositivos, así como herramientas para depuración de memoria y análisis del rendimiento de software. El entorno de desarrollo integrado es Eclipse.
- **Google Play**. Google Play es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un ordenador.
- **Multi-táctil**. Android tiene soporte nativo para pantallas capacitivas con soporte multi-táctil.
- **Videollamada**. Android soporta videollamada a través de Google Talk desde su versión HoneyComb.
- **Multitarea**. La multitarea de aplicaciones está disponible en Android.
- **Tethering**. Android soporta *tethering*, lo que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico.

3.2. Arquitectura de Android

Una de las grandes ventajas de Android es que su arquitectura es de código abierto. De esta forma cualquier persona es capaz de utilizar Android descargando su código completo. A su vez, podemos estudiar su arquitectura de una forma más detallada.

La arquitectura de Android está dividida en capas, como se puede observar en la siguiente figura:

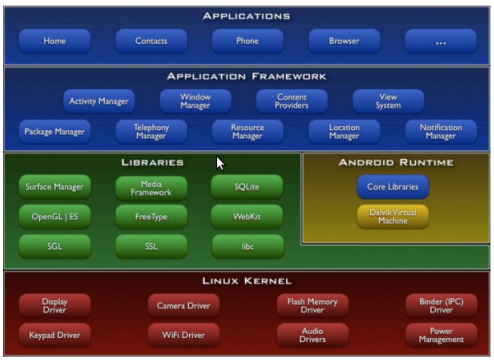

Figura 3.3: Arquitectura de Android

A continuación, se van a estudiar cada una de las capas en detalle:

- **Linux Kernel**. El kernel de Linux ha sido utilizado durante mucho tiempo como un sistema operativo multi-usuario seguro, aislando un usuario del resto. Esta principal característica es la base del sistema de seguridad en Android. Android utiliza esta propiedad como una plataforma multiusuario donde un usuario es una aplicación y cada aplicación está aislada del resto. El kernel de Linux gestiona los drivers de los dispositivos como el bluetooth, la cámara, etc. A su vez, el kernel posee un mecanismo para realizar RPC seguros. Cada aplicación instalada en el dispositivo, posee un único UID (User Identification) y un único GID (Group Identification). El UID es el id de la aplicación durante el tiempo que esté instalada en el dispositivo. Aunque casi todas las aplicaciones están escritas en Java, a veces se require escribir aplicaciones nativas. Dichas aplicaciones son más complejas ya que los desarrolladores necesitan administrar la memoria. Todas las aplicaciones nativas se ajustan al *framework* de Android. Cabe destacar que no hay diferencias en la seguridad de una aplicación nativa y una aplicación Java.
- ***middleware***. Por encima del kernel de Linux se encuentra el *middleware* que proporciona las librerías para el código de ejecución (libSSL, libc, OpenGl, etc). Esta capa también proporciona el entorno de ejecución para aplicaciones Java. Una cuestión importante es: ¿posee Android una máquina virtual de Java para ejecutar las aplicaciones? La respuesta es no. Un fichero java (JAR) no se ejecuta en Android. Lo que proporciona Android es una máquina virtual Dalvik. De esta forma, Android llama a una herramienta llamada dx para convertir código de bytes a ejecutables Dalvik (DEX).
- **Máquina virtual Dalvik**. Dalvik es una máquina virtual de código abierto altamente optimizada y basada en registros. Dalvik está basada en la librería Apache Harmony. Permite ejecutar aplicaciones programadas en Java. No afirma ser una máquina virtual de java pero cumple ese propósito. Dalvik permite crear aplicaciones con un mejor rendimiento y un menor consumo de energía, características

importantísimas en dispositivos móviles. Está optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente. Cada aplicación Java se ejecuta en su propia máquina virtual. Cuando el dispositivo se enciende, un proceso llamado Zygote lanza un proceso de una máquina virtual. Zygote es el encargado de lanzar nuevas máquinas virtuales para los procesos requeridos.

- **Capa de aplicación**. Esta capa es donde residen todas las aplicaciones. Éstas pueden ser tanto aplicaciones de sistema como de usuario. Las aplicaciones del sistema son aquellas que están asociadas al dispositivo como el calendario, el gestor de correo electrónico, los contactos, etc. Los usuarios no pueden desinstalar estas aplicaciones. Las aplicaciones de usuario son las desarrolladas por terceros, que los usuarios instalan o desinstalan en su dispositivo. Incluye interfaces de programación de aplicaciones (APIs) utilizados en las aplicaciones centrales.

Para entender la seguridad en la capa de aplicación es muy importante conocer la estructura de una aplicación Android.

Cada aplicación es creada como una pila de componentes. La ventaja de esta estructura es que cada componente es una entidad auto-contenida en sí misma y que puede ser llamada exclusivamente desde otra aplicación. Este tipo de estructura de aplicación refuerza la compartición de componentes.

La siguiente figura muestra la anatomía de una aplicación Android que consiste en actividades, servicios, broadcast receivers y content providers:

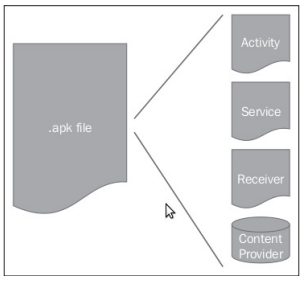

Figura 3.4: Arquitectura de aplicación en Android

Android soporta cuatro tipos de componentes:

- **Actividades**. Este componente es normalmente la interfaz gráfica de la aplicación. Es el componente que interacciona con el usuario. Un ejemplo de una actividad es la página de login donde el usuario introduce sus datos para autenticarse contra el servidor.

- **Servicios**. Este componente se encarga de los procesos que se ejecutan en *background*. Este componente no tiene interfaz gráfica. Un ejemplo podría ser un componente que se sincroniza con el reproductor de audio y con las canciones preseleccionadas por el usuario.

- **Broadcast Receivers**. Es la bandeja de entrada donde se reciben los mensajes del sistema Android o de otras aplicaciones. Como ejemplo, Android lanza un Intent llamado BOOT_COMPLETED después de encenderse.

- **Content Providers**. Este componente es el almacén de datos de la aplicación. La aplicación también puede compartir sus datos con otros componentes de Android. Un ejemplo sería una lista de deseos de una tienda online.

Todos estos componentes están declarados en el AndroidManifest.xml (manifiesto). A su vez, el archivo de manifiesto lista otros requerimientos de aplicación como el mínimo nivel de API requerido, permisos de usuarios, permisos para usar hardware, etc. Más adelante comentaremos estos ficheros a fondo ya que son muy importantes para gestionar de forma correcta la seguridad en los dispositivos Android.

Todos los componentes se comunican entre ellos usando Intents. Los Intents son mecanismos de Android para comunicación asíncrona entre procesos. Hay mecanismos diferenciados para enviar los Intents a cada tipo de componente. Es importante entender que los Intents no son seguros, a menos que estén bien definidos en el manifesto de la aplicación. Cualquier aplicación puede escanear el Intent por lo que no es recomendable poner información sensible en ellos. Y por supuesto, no solo pueden ser escaneados sino que pueden ser alterados por un software malicioso.

En la capa de aplicación, los componentes de Android siguen el modelo basado en permisos. Esto significa que un componente tiene que tener los permisos apropiados para llamar a otros componentes. Aunque Android provee la mayoría de los permisos requeridos para una aplicación, los desarrolladores tienen la habilidad de extender dicho modelo.

3.3. Principales amenazas y mecanismos de seguridad en Android

3.3.1. Amenazas en Android

A la hora de definir una política de seguridad para los dispositivos móviles de la organización, es necesario evaluar los escenarios y las amenazas principales de seguridad que afectan a estos dispositivos hoy en día, y en particular, a la organización bajo estudio. Debe tenerse en cuenta que la movilidad y uso permanente de los dispositivos móviles implica una mayor exposición a amenazas de estos frente a otros dispositivos de la organización (como ordenadores portátiles o de escritorio que sólo se utilizan dentro de la organización), por lo que es necesario disponer de mecanismos de protección adicionales. Los dispositivos móviles son utilizados frecuentemente fuera del control de la organización, especialmente en lo que se refiere a la ausencia de controles de seguridad físicos, salvo los proporcionados por el usuario. Por este motivo, los mecanismos de protección a implantar deben considerar que el dispositivo móvil puede acabar en manos ajenas no autorizadas y deben proteger los datos que almacena y los accesos a servicios remotos de los que dispone.

En primer lugar, sería conveniente definir las principales amenazas en los sistemas generales de seguridad de la información. Para ello, se va a usar la clasificación de Stallings [3]. Stallings agrupa los ataques de la siguiente forma:

- **Intercepción**. El atacante intercepta los mensajes pero no los modifica ni los borra. Esto es un ataque pasivo. Afecta a la privacidad del usuario y al operador de red. El atacante puede usar los datos obtenidos para analizar el tráfico y eliminar la competencia proporcionada por el operador de red.

- **Repetición**. En este caso, el atacante inserta mensajes, datos o lógica de servicio dependiendo del nivel de acceso físico.
- **Modificación de recursos**. El atacante modifica datos, mensajes o lógica de servicio.
- **Denegación de servicios**. En este caso, el atacante intenta sobrecargar la red para negar el servicio a los usuarios.
- **Interrupción**. El atacante causa interrupción destruyendo datos, mensajes o lógica de servicio.

Aunque es una forma muy genérica de definir y agrupar los distintos ataques, sirve para recordar que el objetivo primordial que se debe alcanzar es evitar estos ataques.

Dentro de las amenazas más comunes sobre los dispositivos móviles [4], encontramos frecuentemente, entre otras:

- **Acceso físico no autorizado al dispositivo móvil:**
  - Temporalmente (por un periodo breve).
  - Pérdida o robo del dispositivo móvil (durante un periodo extendido).
- **Acceso no autorizado a la información almacenada:**
  - Datos y documentos corporativos.
  - Credenciales de acceso a servicios corporativos.
- **Acceso no autorizado y manipulación de la información transmitida:**
  - Ataques de hombre en medio (MitM, Man-in-the-Middle).
- **Código móvil malicioso (*malware*) en apps (aplicaciones móviles):**
  - Fraude (servicios SMS Premium), anuncios, privacidad, etc.
  - Uso no autorizado de las capacidades de comunicaciones del dispositivo: NFC, Bluetooth, Wi-Fi, 2/3/4G (SMS, voz y datos), etc.
- **Parches**. Otra de las amenazas más importante es el no llevar nuestro dispositivo actualizado a la última versión de Android. Con las actualizaciones, Android ve reforzados y/o solventados los posibles agujeros de seguridad existentes. Esto influye no sólo en la voluntad del usuario ya que hay muchos dispositivos que no soportan determinadas versiones de Android, lo que supone un riesgo.
- **Teclados**. Los teclados táctiles se han convertido en un elemento prácticamente universal en todos los dispositivos móviles. Hay que tener especial cuidado cuando se teclee información confidencial en lugares públicos ya que nos pueden observar.
- **Seguridad de aplicación**. Muchas aplicaciones móviles son vulnerables a los ataques clásicos como SQL Injection (SQLi), Cross-Site Request Forgery (XSRF), Cross-Site Scripting (XSS), etc. La falta de *sockets* seguros (SSL), la falta de cifrado, etc. son elementos que se echan en falta en las aplicaciones móviles.

3.3.2. Mecanismos de seguridad

Como ya se ha definido anteriormente, Android es un sistema multi-proceso donde cada aplicación se ejecuta en su propio proceso. Casi siempre, la seguridad entre las aplicaciones y el sistema es realizada a nivel de proceso a través de las facilidades del estándar Linux (UID y GID) [5]. Los sistemas operativos Linux utilizan el UID para identificar al usuario particular y el GID para identificar a un grupo. Un usuario puede pertenecer a muchos grupos secundarios y cada grupo secundario poseerá un GID único. Además, el control de acceso es proporcionado a través de un mecanismo de permisos que restringe el acceso a recursos de las aplicaciones. Los mecanismos de seguridad de Android se pueden clasificar en tres grupos: mecanismos de Linux, características del entorno y mecanismos específicos de Android [6].

Mecanismos de Linux

- **POSIX (Portable Operating System Interface)** [7]. Cada apk instalada en Android tiene su propio identificador de usuario de Linux (POSIX). Este ID es asignado cuando la aplicación se instala en el dispositivo. Como consecuencia de ello, el código de dos aplicaciones distintas no puede ejecutarse en el mismo proceso. Esto crea un *sandbox* que evita que una aplicación acceda a los recursos de otra.

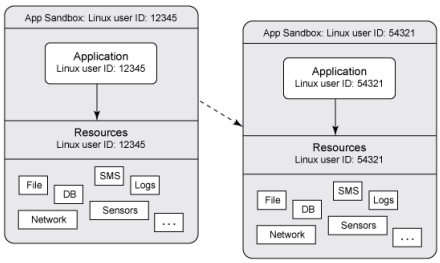

Figura 3.5: Procesos de dos aplicaciones sin compartir recursos

Si una aplicación quiere compartir los mismos recursos que otra e incluso ejecutarse en el mismo proceso, deben compartir el ID. Esto se puede conseguir a través de "sharedUserID" declarado en el manifiesto de la aplicación. También puede ocurrir si dos aplicaciones son firmadas con las misma clave por error. Por ello, hay que destacar la importancia de que los desarrolladores firmen sus aplicaciones de forma segura.

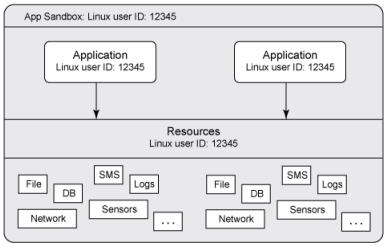

Figura 3.6: Procesos de dos aplicaciones compartiendo recursos

- **Acceso a archivos**. Los ficheros en Android están sujetos a los mecanismos de permisos de Linux. Cada archivo está asociado con su propio usuario, con su GID y con tres tuplas de permisos Read,Write y Execute (rwx) (lectura, escritura y

ejecución). La primera tupla corresponde al propietario; la segunda a los usuarios que pertenecen al grupo; y la tercera al resto de los usuarios. El permiso de accesos deriva del mecanismo de usuarios de Linux y refleja el nivel de acceso permitido a otros usuarios. De la misma forma, los sistemas de archivos están protegidos de otras aplicaciones. Los ficheros creados por una aplicación son asignados al ID de usuario de dicha aplicación y no son accesibles por otras aplicaciones. Un aspecto que refuerza estas medidas de seguridad es que la imagen del sistema esta montada en modo de lectura. Todos los ejecutables y archivos de configuración importantes están localizados en RAM (que es solamente de lectura pero se reinicia en cada encendido del dispositivo) o en la imagen del sistema. Por lo tanto, un atacante que obtenga la habilidad de escribir en ficheros en cualquier parte del sistema de archivos, no obtiene los permisos necesarios de remplazar archivos críticos. No obstante, un atacante podría montar la imagen del sistema siempre y cuando tenga acceso de superusuario (root). Otros dos aspectos importantes en el sistema de archivos de Android son la tarjeta SD y la partición de datos. La partición de datos es el espacio donde todos los datos de usuarios y aplicaciones son almacenados. Esto previene al sistema de daños en caso que el usuario (o un atacante) instale demasiadas aplicaciones o cree demasiados archivos. Cuando un dispositivo Android es ejecutado en modo seguro, los datos de la partición de datos no son cargados, haciendo posible recuperarse de este tipo de ataques. La tarjeta SD puede ser manipulada off-line y fuera del control del dispositivo ya que es almacenamiento externo.

Características del entorno

- **Unidad de Gestión de Memoria (MMU)**. Un prerrequisito para muchos de los sistemas operativos modernos, y para Linux en particular, es la Unidad de Gestión de Memoria, un componente hardware que facilita la separación de los procesos de diferentes espacios de direcciones de memoria virtual. Varios sistemas operativos emplean la MMU de tal forma que un proceso es incapaz de leer las páginas de memoria de otro proceso o de corromper su memoria.

- **Seguridad de tipos**. Es una propiedad de los lenguajes de programación modernos. Fuerza al contenido de una variable a corresponderse con un formato específico y así prevenir usos erróneos. Sin esta característica se puede llegar a corromper la memoria y a realizar ataques de overflow. Java es un lenguaje fuertemente tipado, por lo que estos ataques son evitados. No obstante, Android permite componentes escritos en C, con el riesgo de reducir la seguridad. Binder, el IPC específico de Android, también tiene seguridad de tipos [8].

- **Características de seguridad móvil**. Un set básico de atributos de la telefonía móvil viene de la necesidad de identificar al usuario, monitorizar su uso y tarifar al cliente de forma correcta. Un término más general es AAA (Authentication, Authorization y Accounting). Android obtiene estos clásicos de la seguridad a través del diseño del dispositivo móvil.

Mecanismos de seguridad específicos

- **Permisos de aplicación**. El corazón de la seguridad en el nivel de aplicación es el sistema de permisos, el cual hace cumplir las restricciones necesarias en cada operación que la aplicación puede realizar. El Package Manager se encarga de otor-

gar permisos a las aplicaciones en la instalación mientras que el *framework* de la aplicación se encarga de aplicar los permisos del sistema en la ejecución.

Existen alrededor de 100 permisos en Android, los cuales controlan operaciones como: el marcado del teléfono, hacer fotografías, usar Internet, escribir SMS, etc.

Cualquier aplicación Android puede declarar permisos adicionales. Para obtener estos permisos, la aplicación debe solicitarlo explícitamente en su manifiesto.

Los permisos llevan niveles de protección asociados:

1. Normal. Permisos que no son especialmente peligrosos si se obtienen.
2. Peligrosos (Dangerous). Permisos más peligrosos que los de tipo 1 o normalmente no solicitados por las aplicaciones. Estos permisos necesitan la confirmación por parte del usuario.
3. Firma (Signature). Permisos que solo pueden ser otorgados a otros paquetes que se han firmado con la misma firma que la declarada en el permiso.
4. SignatureOrSystem. Permisos de firma que también se conceden a los paquetes instalados en la imagen del sistema.

La asignación del nivel de protección se deja a la voluntad de los desarrolladores. No obstante, las siguientes directrices son útiles a la hora de decidir el nivel de protección de una aplicación. Los permisos "Normal" deben implicar un riesgo menor y servir únicamente como una llamada de atención al usuario para que se de cuente de que la aplicación requiere acceso a una funcionalidad concreta. Los permisos "Dangerous" deben ser usados para operaciones que implican un riesgo mayor. Un equilibrio balanceado debe mantener la diferencia entre estas dos categorías para evitar desclasificar operaciones de riesgo como normales, lo que provocaría la concesión equivocada de privilegios. Mientras que si optamos por declarar todos los permisos como "Dangerous", es igual de peligroso ya que puede hacer que el usuario ignore la importancia del nivel de protección debido a la falta de contraste.

Los permisos de "Signature" están destinados únicamente a las aplicaciones firmadas por el mismo desarrollador.

El proceso que siguen los permisos es el siguiente: en la instalación, los permisos requeridos por la aplicación son concedidos basándose en comprobaciones con la firma de las aplicaciones que declaran esos permisos y la interacción con el usuario. Una vez la aplicación se ha instalado y sus permisos han sido concedidos, no se pueden solicitar más permisos. Una operación que no tenga permisos fallará en el momento de su ejecución. Por lo tanto, a la hora de instalar una aplicación existen dos opciones: confiar en el desarrollador o no confiar en él. En caso de no confiar en él se recomienda no instalar la aplicación ya que no funcionará de forma correcta.

Además de proteger las APIs, el mecanismo de permisos debe garantizar la seguridad de varios componentes en una aplicación: Activity, Service, Content Provider y Broadcast Receivers. Esto se consigue asociando permisos con el componente relevante en su declaración y en el manifiesto.

Una Activity (cualquier pantalla con la que el usuario pueda interactuar) puede especificar una serie de permisos requeridos para cualquier aplicación que desea lanzarla. De forma similar, un Service (estructura de fondo de la aplicación) puede controlar qué aplicaciones están permitidas a enlazarse con él. El Content Provider (encargado de almacenar y compartir datos) define permisos para regular quién tiene autorización para escribir o leer información. Como los permisos de lectura y escritura están definidos de forma individual y no están interconectados, se puede tener buen control sobre el Content Provider.

Los permisos sobre los Broadcast Receivers (ejecutados por el sistema en reacción a los Intents) hacen posible controlar qué componentes se pueden recibir. La implicación de estos permisos es tal que cualquier intento de espiar los Intents que están siendo enviados es evitado y cualquier esfuerzo para enviar componentes no autorizados fallará.

- **Encapsulación de componentes**. La habilidad de encapsular componentes en una aplicación evita cualquier acceso a ella desde otra aplicación (asumiendo que tiene un ID de usuario distinto).

  Esto se consigue fundamentalmente definiendo el atributo ”exported” del componente. Si este atributo está marcado como ”false”, el componente solo puede ser accedido por el propietario de la aplicación y por otras aplicaciones que compartan el id de usuario con el atributo ”shared user-ID”. Si está marcado como ”true”, puede ser invocado por otras aplicaciones externas.

  Sin embargo, las aplicaciones invocadoras todavía pueden ser controladas con el mecanismo de permisos explicado anteriormente. No obstante, se recomienda marcar este atributo manualmente y no confiar en el comportamiento por defecto del sistema.

- **Firma de aplicaciones**. Cada aplicación en Android está empaquetada en un fichero .apk. Este fichero es similar al estándar Java (.jar). A su vez, el .apk también incluye todos los recursos necesarios (como las imágenes). Android requiere que todas las aplicaciones sean firmadas digitalmente[9]. La validez de la firma es correcta mientras que su certificado sea valido y la clave pública adjunta verifique dicha firma.

  La firma sirve para verificar que dos o más aplicaciones son del mismo propietario. Esta característica es usada por el mecanismo ”sharedUserID” y por el mecanismo de permisos (Signature y SignatureOrSystem).

3.4. ADB

Android Debug Bridge (adb) es una herramienta que se utiliza para interactuar con nuestro *smartphone* de una forma más avanzada y completa. Es una línea de comandos que permite comunicarse con el dispositivo Android a través de una instancia emulada o de un dispositivo conectado. Es un programa cliente-servidor que incluye:

- **Cliente**. Corre en la máquina de desarrollo. Se puede invocar un cliente desde la consola usando un comando adb.
- **Servidor**. Corre un proceso en *background* en la máquina de desarrollo.
- **Demonio**. Corre un proceso en *background* en el dispositivo conectado.

Los comandos básicos de adb son:

- **adb devices**. Imprime un listado con todos los dispositivos conectados al ordenador.
- **adb push**. Permite extraer un archivo del teléfono y pasarlo al ordenador.
- **adb install**. Permite instalar aplicaciones en el dispositivo.
- **adb shell**. Entra en un intérprete de comandos dentro del dispositivo Android.
- **adb reboot**. Reinicia el dispositivo.

En este proyecto, se usará esta herramienta para configuraciones y modificaciones avanzadas sobre el dispositivo Android.

3.5. Android en entornos empresariales

Una de las principales tendencias de la industria móvil durante los últimos años es BYOD (Bring Your Own Device), es decir, un escenario en el que se permite a los usuarios y empleados de la organización hacer uso de sus dispositivos móviles personales para el acceso al entorno, servicios y datos corporativos [10].

Esta tendencia tuvo una aceptación inicial muy alta, principalmente en los departamentos financieros de las organizaciones, debido al ahorro de costes asociado a la adquisición inicial de los dispositivos móviles, frente a la opción de que los dispositivos móviles sean subvencionados por la organización y proporcionados a los usuarios como una herramienta de trabajo más.

Debido a que los dispositivos móviles actuales son sistemas complejos y avanzados, su precio es bastante elevado, hecho que tiene influencia especialmente en las decisiones de la organización cuando es necesario adquirir un número elevado de éstos (decenas, cientos o miles de dispositivos móviles).

El hecho de que la organización no tenga que hacer ese desembolso inicial, y el coste del dispositivo móvil sea financiado por el usuario, al ser este su propietario en lugar de la organización, conllevó una reducción de costes inmediata para muchas organizaciones.

Sin embargo, es necesario tener en cuenta que un entorno BYOD más heterogéneo conlleva una serie de gastos e inversiones adicionales de gestión, mantenimiento e integración de los dispositivos móviles en los entornos TIC de las organizaciones, que normalmente no es tenido en cuenta.

La ventaja principal de BYOD para el usuario es que éste puede hacer uso libremente del dispositivo móvil que mejor se adapta a él, qué más le gusta o con el que está más familiarizado.

La desventaja principal de BYOD para la organización es que tiene que acomodar las dos facetas del dispositivo móvil, personal y profesional, en el entorno TIC y de manera segura, así como definir y conjugar los requisitos y necesidades de ambos mundos.

La tendencia BYOD debe asumir que los dispositivos móviles personales no son de confianza desde el punto de vista de seguridad, salvo que hayan sido configurados con unos requisitos mínimos de seguridad y pasen a ser gestionados y monitorizados por la organización.

Una aproximación intermedia adoptada por algunas organizaciones conocida como CYOD (Choose Your Own Device), con el objetivo principal de intentar obtener lo mejor de ambos mundos, es que el usuario pueda elegir el dispositivo móvil que utilizará, pero en lugar de hacerlo de todos los disponibles en el mercado, únicamente lo seleccionará de una reducida lista preaprobada de dispositivos móviles que cumplen con los requisitos de seguridad y tecnológicos de la organización.

Estas aproximaciones complementan el escenario dónde el dispositivo móvil es propie-

dad de la organización y sólo puede ser empleado para tareas profesionales, o un escenario intermedio, conocido como COPE (Corporataly-Owned, Personally-Enabled), donde el dispositivo móvil es propiedad de la organización pero se permite su utilización para ciertas tareas personales

Finalmente, debe considerarse también otro modelo de uso reciente en la industria basado en la posibilidad de compartir un dispositivo móvil entre diferentes empleados de la organización, debiendo la solución MDM proporcionar capacidades multiusuario para permitir y segmentar el acceso a los datos y servicios a cada uno de los usuarios, o en su defecto, permitir una rápida y segura transición del dispositivo móvil al ser transferido de un usuario a otro.

La definición de una política de seguridad para los dispositivos móviles, mencionada previamente como requisito básico antes de la adopción de una solución MDM, se hace aún más patente en entornos BYOD, siendo necesario que la política diferencie entre los requisitos asociados a los dispositivos de la organización y los dispositivos móviles personales empleados en el entorno corporativo.

Existen numerosas implicaciones legales asociadas al uso de dispositivos móviles, y los datos que éstos gestionan, en entornos BYOD, debido al carácter personal de los mismos, que quedan fuera del alcance del presente documento.

A modo de referencia, algunos de los aspectos legales a considerar (especialmente si no se dispone de consentimiento del usuario por escrito) son: la monitorización de los dispositivos móviles y su tráfico de red para detectar violaciones en la política de seguridad de la organización, el borrado de datos apps personales, el acceso a datos personales almacenados en el dispositivo móvil o en servicios ”en la nube” durante las auditorías o la investigación de un incidente de seguridad, la monitorización de los dispositivos móviles fuera de las dependencias de la organización, como por ejemplo mediante la utilización de mecanismos de localización del dispositivo móvil para conocer su ubicación física, o la responsabilidad a la hora de compensar, reparar o sustituir el dispositivo móvil usado por haber sido robado, perdido o dañado, así como las implicaciones legales de la distribución o uso de software pirata o sin licencia y *malware*.

Los aspectos legales son condición suficiente en muchas organizaciones para mantener la propiedad de los dispositivos móviles que son entregados a los usuarios y facilitar así las tareas de monitorización, el borrado remoto de datos o la realización de auditorías de seguridad sobre los mismos, manteniendo en todo momento el control sobre la propiedad intelectual y los derechos de autor de los contenidos por parte de la organización.

Se recomienda por tanto consultar al departamento legal de la organización para analizar y profundizar en estos aspectos, que habitualmente tienen implicaciones directas sobre la privacidad del usuario.

Por otro lado, una de las características deseadas de las soluciones MDM para entornos BYOD, y potencialmente también para entornos únicamente corporativos, es la posibilidad de definir el número de dispositivos móviles gestionados permitidos por usuario.

En el caso de implantar un entorno BYOD, se recomienda disponer (a través de un servidor web interno) de tutoriales, vídeos y documentación qué permitan a los empleados de la organización llevar a cabo el registro de sus dispositivos móviles personales en la solución MDM de manera sencilla y con la menor implicación posible por parte de los administradores TIC.

Aunque la industria ha adoptado de forma general el término BYOD (Bring Your Own Device) para referenciar este escenario de uso, el término correcto y que refleja el riesgo para las organizaciones debería ser CYOD (Connect Your Own Device), ya que no existe riesgo de seguridad asociado hasta que el usuario conecta su dispositivo móvil personal a la red (por ejemplo, vía Wi-Fi) o sistemas (por ejemplo, vía USB) de la organización para acceder a los servicios y datos corporativos.

Debido al dinamismo y continua evolución de la industria móvil, recientemente ha surgido otra tendencia que está siendo adoptada ampliamente por numerosas organizaciones, denominada BYOA (Bring Your Own App(lication)).

En escenarios BYOA los usuarios instalan apps de los mercados de aplicaciones públicos en sus dispositivos móviles personales o corporativos para incrementar su productividad, empleándolas para realizar sus tareas profesionales y manejar, por tanto, datos corporativos.

Para acomodar todos estos escenarios de uso, una tendencia más reciente y práctica, desde el punto de vista empresarial, es clasificar los dispositivos móviles en dos grandes grupos, dispositivos gestionados y no gestionados, independientemente de quién es el propietario del propio dispositivo móvil.

Los dispositivos gestionados son aquellos que han seguido el proceso de registro o enrollment, y por tanto, sobre los que se ha aplicado la política de seguridad de la organización y la configuración adecuada a los requisitos de seguridad del dispositivo y de la información que maneja. Estos dispositivos están englobados y gestionados por la solución MDM.

Por el contrario, los dispositivos no gestionados son aquellos sobre los que la organización no ha llevado las tareas de configuración previamente definidas, y por tanto, no dispone de control sobre su configuración o el nivel de seguridad de los mismos.

Esta clasificación puede ser extendida igualmente a las aplicaciones móviles, disponiendo entonces de dos tipos de apps, apps gestionadas frente a no gestionadas.

En la siguiente tabla se muestran los mecanismos de seguridad a nivel empresarial ofrecidos por Android en sus distintas versiones:

Froyo (release 2.2)	Password policy. Remote wipe. Remote locks.
Gingerbread (release 2.3)	SIP support.
Honeycomb (release 3.0)	Encryption and password policy for tablets. System encryption for tablets.
Ice Cream Sandwich (release 4.0)	Extended system encryption, encryption and password policies to devices. Certificate management capabilities. VPN. Developer interfaces for SSL VPN. Facial recognition unlock.Network data usage monitoring. Offline email searching.
Kit Kat (release 4.4)	Buffer overflow detection. Peruser VPN for shared devices.

Mecanismos de seguridad a nivel empresarial de Android

Manifiestos en Android

En todas las aplicaciones Android el conjunto de componentes de la aplicación y los módulos de seguridad son almacenados en un fichero que define la política de la aplicación en cuestión. Este fichero de política se llama "AndroidManifest.xml" y es el fichero más importante de la aplicación. En este fichero se define la política de control de acceso para la aplicación y sus componentes. A su vez, declara los componentes de la aplicación, su visibilidad, sus reglas de acceso, librerías, características y la versión mínima de Android necesaria para ejecutar la aplicación.

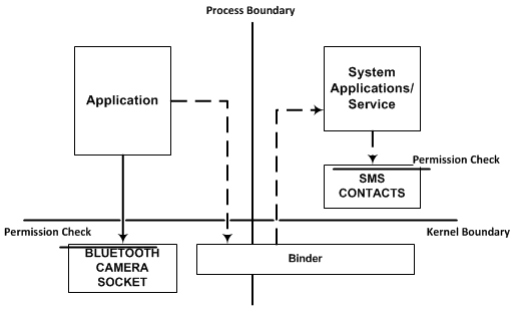

Figura 3.7: Flujo de los permisos en una apk

El fichero de manifiesto no es extensible, por lo que las aplicaciones no pueden añadir sus propios atributos o etiquetas. Hay dos etiquetas imprescindibles en estos ficheros: $< manifest >$ y $< application >$.

La etiqueta $< manifest >$ declara los atributos específicos de la aplicación. Los atributos de esta etiqueta son:

- **package**. Es el nombre del paquete. Es el namespace de Java, por ejemplo com.Android.example.
- **Android:sharedUserId**. Este atributo es usado si dos o más aplicaciones comparten el mismo Linux ID.
- **Android:sharedUserLabel**. Es el nombre de usuario del User ID compartido y sólo tiene sentido si sharedUserId está activo.
- **Android:versionCode**. Es la versión de código utilizado internamente. Sirve para realizar revisiones de código.
- **Android:versionName**. Es la versión de la aplicación mostrada al usuario.
- **Android:installLocation**. Este atributo define la localización de la instalación del fichero .apk.

Los atributos de la etiqueta $< application >$ son:

- **Android:allowTaskReparenting**. Este valor puede ser sobrescrito por el valor de $< Activity >$. Sirve para permitir a una actividad ser lanzada una vez más desde su actividad padre.
- **Android:*backup*Agent**. Contiene el nombre del agente de *backup* de la aplicación.

- **Android:debuggable**. Este atributo indica si una aplicación puede ser *debugueada* o no. Antes de lanzar una aplicación al market se debe poner a false.
- **Android:description**. Es la descripción de la aplicación mostrada al usuario.
- **Android:enabled**. Si este atributo está puesto a true, el sistema Android puede instanciar componentes de aplicación.
- **Android:hasCode**. Si el atributo está puesto a true, la aplicación intentará cargar código cuando lance los componentes.
- **Android:hardwareAccelerated**. Este atributo permite a la aplicación soportar un rendering de aceleración hardware.
- **Android:icon**. Es una referencia al icono de la aplicación.
- **Android:killAfterRestore**. Si este atributo está puesto a true, la aplicación será finalizada después de una restauración completa del sistema.
- **Android:largeHeap**. Este atributo permite a Android crear una gran pila Dalvik para la aplicación e incrementar la memoria de la aplicación.
- **Android:label**. Es la etiqueta mostrada al usuario.
- **Android:logo**. Es el logo de la aplicación.
- **Android:manageSpaceActivity**. Es el nombre de la actividad que gestiona la memoria de la aplicación.
- **Android:name**. Este atributo contiene el nombre de la actividad que será instanciada antes de que algún componente se lance.
- **Android:permisión**. Asigna el permiso que un cliente debe tener con la aplicación para poder interactuar con ella.
- **Android:persistent**. Permite a la aplicación estar lanzada continuamente.
- **Android:process**. Este atributo es el nombre del proceso en el cual un componente se ejecutará.
- **Android:restoreAnyVersion**. Este atributo permite al *backup*Agent realizar el *backup* desde cualquier versión.
- **Android:supportsRtl**. Cuando este atributo está puesto a true, permite *layouts* de derecha a izquierda.
- **Android:taskAffinity**. Este atributo permite tener a todas las actividades afinidad con el nombre del paquete.
- **Android:theme**. Es una referencia al estilo de la aplicación.
- **Android:uiOptions**. Este atributo sirve para indicar opciones especiales de la interfaz gráfica.

Otra de las partes más importantes del fichero de manifiesto es la definición de los permisos. Estos se declaran usando la etiqueta $< uses - permission >$. Estos permisos son mostrados al usuario cuando instalan una aplicación. Cabe destacar que cuando un usuario acepta los permisos en la instalación de una aplicación, no puede rechazarlos posteriormente. Sería necesario desinstalar la aplicación por completo.

A la hora de analizar los manifiestos de las aplicaciones, existen diversas herramientas que realizan esta función. Dichas herramientas realizan un análisis estático o dinámico de los programas en cuestión.

El análisis estático de software es un tipo de análisis de software que se realiza sin ejecutar el programa. El análisis se realiza en alguna versión del código fuente o sobre el código objeto.

Mientras tanto, el análisis dinámico es un tipo de análisis de software que supone la ejecución del programa y la observación de su comportamiento. Para que dicho análisis resulte efectivo, el programa a ser analizado se debe ejecutar con los suficientes casos de prueba como para producir un comportamiento de interés.

A continuación se van a estudiar dos herramientas de análisis de software: TaintDroid (análisis dinámico) y Comdroid (análisis estático).

- **TaintDroid** [11]. TaintDroid es un nuevo *tester* de aplicaciones para dispositivos Android que fue desarrollado por investigadores de universidades estadounidenses. Concretamente en Duke y Penn State crearon un programa de seguimiento para ver qué hacen las aplicaciones Android de fondo. El resultado fue el esperado, aproximadamente más de la tercera parte de las aplicaciones gratuitas para dispositivos Android que se pueden descargar en Android Market, extraen información con objetivos de recabar datos para fines publicitarios. No se trata de virus ni troyanos propiamente dichos, pero sí del estudio del comportamiento de altas cifras de personas que utilizan su teléfono móvil. Entre los datos emitidos por estas aplicaciones está el envío de coordenadas GPS y el número de teléfono, entre otros.

  TaintDroid, es una extensión muy inteligente que incluso puede llegar a renovar el concepto de los sistemas de protección de datos privados porque permite ver a los usuarios lo que las apps que han descargado están haciendo momento a momento gracias al uso de una versión similar al Dalvik VM y un módulo de Kernel que intercepta la actividad del sistema en tiempo real. Cuando la aplicación comienza el proceso de envío de datos privados a una red externa, aparece una ventana emergente que advierte al usuario de tal maniobra.

- **ComDroid**. ComDroid es un software gratuito que permite analizar las vulnerabilidades existentes en una apk [12]. Fundamentalmente, está diseñada para que los desarrolladores de software analicen sus aplicaciones antes de lanzarlas al mercado. No obstante, también es útil para analizar aplicaciones. En este documento, como ejemplo de su uso, se va a analizar una aplicación de mi autoría desarrollada para la asignatura "Desarrollo de aplicaciones para dispositivos móviles". (ver Fig.3.8)

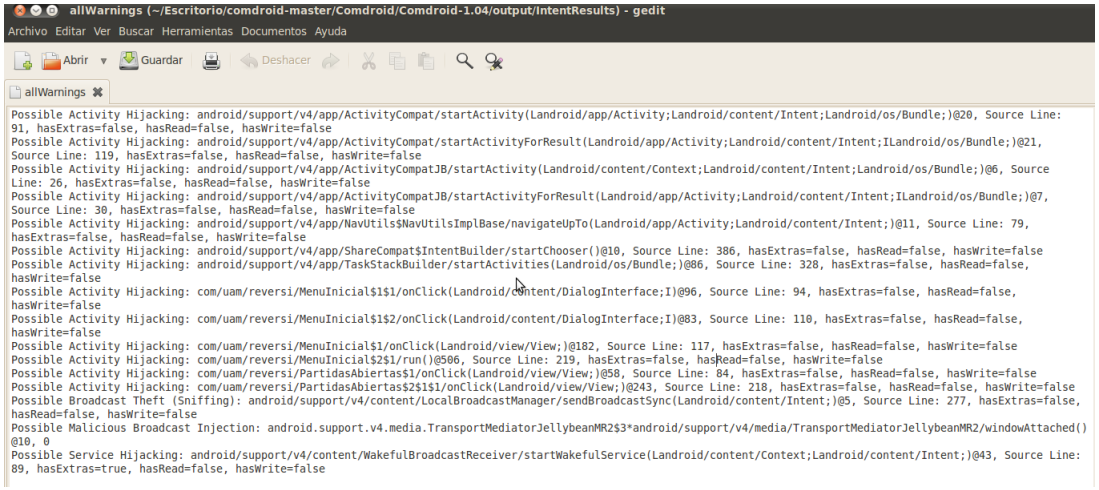

Figura 3.8: Ejemplo de ComDroid

Como se puede observar en la imagen, la aplicación analizada tiene ciertos riesgos de ser vulnerada. De forma muy detallada indica la clase, incluso la línea donde se ha cometido un fallo de seguridad. Por ello, podemos afirmar que es una herramienta muy útil a la hora de analizar las aplicaciones antes de lanzarlas al mercado.

Como ejemplo del funcionamiento de los permisos y manifiestos en Android, se ha creado una aplicación muy simple que inicia la cámara del dispositivo siempre y cuando en el manifiesto de la aplicación se indique. Para ello, el permiso necesario para acceder a la cámara sería:

```
<uses-permission Android:name="Android.permission.CAMERA" />
```

A continuación, se muestra la respuesta de la aplicación tanto en el caso de tener el permiso como en caso contrario:

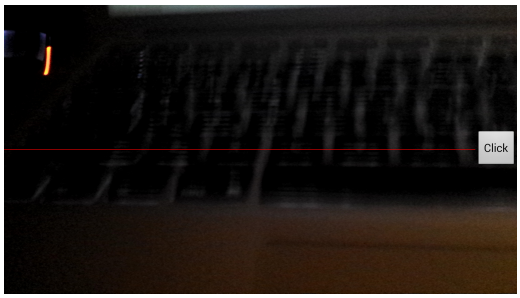

Figura 3.9: Ejemplo de acceso a la cámara con permisos

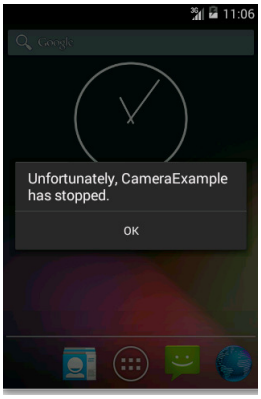

Figura 3.10: Ejemplo de acceso a la cámara sin permisos

Capítulo 4

Implementación de la política de seguridad

En este capítulo del documento se pretende realizar una implementación de una política de seguridad para dispositivos Android. Para conseguir este objetivo, se ha realizado un estudio de diversas herramientas existentes, que en su conjunto conforman un entorno más que interesante para la implementación en una entidad corporativa [13].

A su vez, abogando por el ahorro y la economía de la empresa, se ha realizado un esfuerzo por facilitar herramientas de código abierto. Esto se debe a que no existen costes de licencias, ni de mantenimientos, ni de soporte. En principio, su coste se reduciría a la instalación, configuración y formación. Otra de las grandes ventajas que aporta el open-source es su facilidad de personalización. Esto quiere decir que si, en un momento dado, el administrador de seguridad de la empresa desea incorporar una funcionalidad nueva a la herramienta que esta no posee, es tan sencillo como incluirlo en el mismo código fuente de la herramienta, ya que se tiene acceso al mismo.

Para realizar las pruebas, se va a utilizar un dispositivo Samsung Galaxy S4 y un Samsung Galaxy Tab 3 con permisos de super-usuario. Para obtener estos permisos, hay que realizar un proceso llamado *rooteo*.

A lo largo de este capítulo, se realizará el *rooteo* del dispositivo con el que poder obtener permisos de superusuario. A su vez, gracias al *rooteo* se podrá gestionar de forma eficiente y segura la conectividad y el almacenamiento en el dispositivo. En estas secciones se tratarán aspectos como canales SSH, conexiones VPN y deshabilitación de la instalación de software.

4.1. Obtención de permisos de superusuario: rooteo de un dispotivo Android

Rootear un dispositivo es cambiar el estado de usuario regular a usuario con privilegios y ventajas, mediante alguna de las aplicaciones desarrolladas para ello. Generalmente, el propietario de un dispositivo lo *rootea* para tomar el control del mismo. Esto significa eliminar las limitaciones que fabricantes y operadoras agregan a los equipos y, por tanto, tener acceso y control sobre todas las funciones del dispositivo.

Son muchas las ventajas de ser super-usuario de un dispositivo, pero la más valorada es que otorga la posibilidad de personalizarlo completamente mediante la instalación de ROM (versiones de Android distintas a las que viene por defecto). Otra de las ventajas es la posibilidad de eliminar las aplicaciones que incluyen los fabricantes, porque consumen memoria del dispositivo y muchas de ellas no se utilizan.

No obstante, no siempre es recomendable *rootear* el dispositivo Android, ya que hay que tener especial cuidado con ciertos aspectos. El aspecto más importante es el riesgo que corre la seguridad de nuestros datos. Tener permisos de administrador es equivalente a tener acceso completo a nuestro teléfono. Este privilegio no solamente es nuestro, sino también de las aplicaciones que se instalan y, si son peligrosas o están infectadas, pueden tener acceso a la información que deseen.

A su vez, el *rooteo* hace que el teléfono sea más inestable. De hecho, el root no tiene ningún efecto negativo en el sistema Android pero, en general, si se realizan modificaciones al dispositivo (instalaciones especiales, personalizaciones pesadas, etc) incidirá a que el sistema no sea tan estable.

Realizar el *rooteorooteo* es bastante sencillo. Únicamente hay que descargar el programa Odín, un Custom Recovery y un paquete raíz que permita hacer la escalada de privilegios. En primer lugar, es necesario copiar el paquete raíz a nuestro dispositivo. Posteriormente, cargamos el Custom Recovery mediante Odín al dispositivo. Y por último, desde el nuevo Custom Recovery ejecutamos el paquete raíz y realizamos de forma automática el *rooteo*.

4.2. Configuración de las interfaces de comunicación de un dispositivo Android

Los dispositivos móviles existentes a día de hoy poseen una gran cantidad de interfaces de comunicación. Cada interfaz de comunicación implica una posible vulnerabilidad en el dispositivo debido a la transmisión de datos. En este capitulo se analizarán las interfaces de comunicación más importantes y se intentarán eliminar las posibles vulnerabilidades existentes.

4.2.1. SSH

Rootear un dispositivo Android activa el acceso a una cantidad enorme de aplicaciones que no pueden ser instaladas sin ser super-usuario y que permitirán configurar todos y cada uno de los parámetros del dispositivos. En nuestro caso en particular, el ser super-usuario nos ha permitido realizar tareas que no es capaz de hacer un usuario normal y establecer un canal SSH para que el administrador de sistemas pueda acceder al dispositivo de forma remota.

Secure Shell o SSH es un protocolo de red que permite el intercambio de datos sobre un canal seguro entre dos computadoras. SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión. SSH usa criptografía de clave pública para autenticar el equipo remoto y permitir al mismo autenticar al usuario si es necesario. (ver Fig.4.1)

Además de la conexión a otros dispositivos, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

Figura 4.1: Esquema SSH

El protocolo SSH proporciona los siguientes tipos de protección:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.
- El cliente transmite su información de autenticación al servidor usando un cifrado robusta de 128 bits.
- Todos los datos enviados y recibidos durante la sesión se transfieren por medio de cifrado de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer.

En el contexto empresarial sobre el cual está enfocado este trabajo, la figura del administrador de sistemas de una empresa es muy importante. Gracias a SSH, el administrador puede gestionar de forma remota todos los parámetros de configuración del dispositivo Android. A través de una conexión SSH se podrá des/habilitar instalaciones de software, la tarjeta SD, el bluetooth, etc.

Para posibilitar la conexión mediante SSH, se ha instalado en el dispositivo la aplicación ”SSHDroid”, la cual necesita permisos de root. En dicha aplicación configuramos la contraseña que queremos utilizar, así como la dirección IP y el puerto del *host*. Para poder conectarnos al dispositivo, simplemente es necesario abrir una terminal y teclear, a modo de ejemplo: SSH -p 2222 root@192.168.1.49 . En este punto ya estaremos conectados y seremos capaces de gestionar los parámetros deseados, como se verá más adelante.

4.2.2. VPN

¿Qué es una VPN?

Para poder habilitar redes privadas distribuidas y comunicar de forma segura cada uno de los nodos de una red pública hay una necesidad de evitar que los datos sean interceptados. Con una Red Privada Virtual (VPN), los usuarios remotos que pertenecen a una red privada pueden comunicarse de forma libre y segura entre redes remotas a través de redes públicas [14].

Una VPN consiste en varias máquinas (cada una en cada ”extremo” de la conexión) y una ruta o ”túnel” que se crea dinámicamente en una red pública o privada. Para asegurar la privacidad de esta conexión los datos transmitidos entre ambos ordenadores son cifrados por el Point-to-Point Protocol, también conocido como PPP, un protocolo de acceso remoto, y posteriormente enrutados o encaminados sobre una conexión previa (también remota, LAN o WAN) por un dispositivo PPTP (ver Fig.4.2).

Una VPN normalmente usa la red Internet como transporte para establecer enlaces seguros, extendiendo las comunicaciones a oficinas aisladas. Significativamente, decrece el coste de las comunicaciones porque el acceso a Internet es generalmente local y mucho más barato que las conexiones mediante Acceso Remoto a Servidores.

Una Red Privada Virtual (VPN) transporta de manera segura por Internet a través de un túnel establecido entre dos puntos que negocian un esquema de cifrado y autentificación para el transporte. Una VPN permite el acceso remoto a servicios de red de forma transparente y segura con el grado de conveniencia y seguridad que los usuarios conectados elijan.

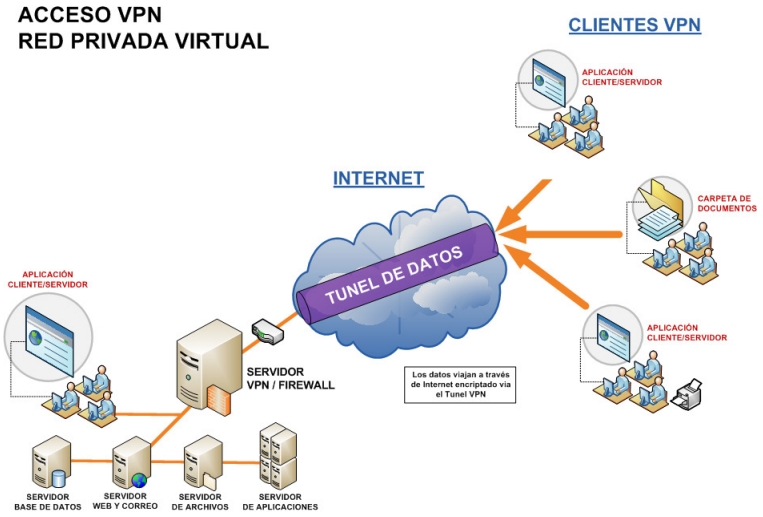

Figura 4.2: Esquema VPN

Una VPN es una red de datos de gran seguridad que permite la transmisión de información confidencial entre la empresa y sus sucursales, socios, proveedores, distribuidores, empleados y clientes, utilizando Internet como medio de transmisión. Aunque Internet es una red pública y abierta, la transmisión de los datos se realiza a través de la creación de túneles virtuales, asegurando la confidencialidad e integridad de los datos transmitidos.

Las VPNs permiten:

- La administración y ampliación de la red corporativa al mejor costo-beneficio.
- La facilidad y seguridad para los usuarios remotos de conectarse a las redes corporativas.

Los requisitos indispensables para esta interconectividad son:

- Políticas de seguridad.

- Requerimiento de aplicaciones en tiempo real.
- Compartir datos, aplicaciones y recursos.
- Servidor de acceso y autentificación.
- Aplicación de autentificación.

Protocolos de VPN

Durante largo tiempo han sido implementados varios protocolos de red para el uso de las VPN. Estos protocolos intentan cerrar todos los agujeros de seguridad inherentes en VPN.

Dichos protocolos son los siguientes:

- **Point-to-Point Tunneling Protocol (PPTP)**. PPTP es una especificación de protocolo desarrollada por varias compañías. Normalmente, se asocia PPTP con Microsoft, ya que Windows incluye soporte para este protocolo. Los inicios de PPTP para Windows contenían características de seguridad demasiado débiles para usos en contextos empresariales. Por eso, Microsoft continúa mejorando el soporte PPTP. La mejor característica de PPTP radica en su habilidad para soportar protocolos no IP. Sin embargo, el principal inconveniente de PPTP es disponer de sólo un único cifrado y autentificación estándar: dos productos que se usan bajo PPTP pueden llegar a ser completamente incompatibles simplemente porque el cifrado de los datos sea diferente.
- **Layer Two Tunneling Protocol (L2TP)**. El principal competidor de PPTP en soluciones VPN fue L2F, desarrollado por Cisco. Con el fin de mejorar L2F, se combinaron las mejores características de PPTP y L2F para crear un nuevo estándar llamado L2TP. L2TP existe en el nivel de enlace del modelo OSI. L2TP, al igual que PPTP soporta clientes no IP, pero también da problemas al definir un cifrado estándar.
- **Internet Protocol Security (IPsec)**. IPsec es en realidad una colección de múltiples protocolos relacionados. Puede ser usado como una solución completa de protocolo VPN o simplemente como un esquema de cifrado para L2TP o PPTP. IPsec existe en el nivel de red en OSI, para extender IP con el propósito de soportar servicios más seguros basados en Internet.
- **SOCKS Networks Security Protocol**. El sistema SOCKS proporciona otra alternativa a los protocolos de VPN. SOCKS se aloja en el nivel de sesión de OSI. Como SOCKS trabaja en un nivel OSI más alto que los protocolos anteriores, permite a los administradores limitar el tráfico VPN.

Implementación de una VPN

Para la implementación de una VPN se va a utilizar OpenVPN [15].

OpenVPN es una solución multiplataforma de código abierto que ha simplificado la configuración de VPNs frente a otras soluciones más antiguas y difíciles de configurar,

haciéndola más accesible para usuarios inexpertos en este tipo de tecnología. Es un software basado en autenticación SSL/TLS, que implementa conexiones de capa 2 o capa 3 del modelo OSI. Esto da cuenta de la gran flexibilidad del proyecto ya que no todos los protocolos lo admiten. Entre las cualidades más destacadas encontramos la capacidad de trabajar bajo red NAT. Es decir, nuestros servidores y clientes pueden estar trabajando con IPs privadas y funcionar perfectamente. En cuanto a las conexiones soportadas, pueden existir redes con Proxy sin inconvenientes, además de trabajar con IPs dinámicas asociándolas a un cliente DDNS. Por lo tanto, si tenemos conexiones de internet convencionales como ADSL, OpenVPN se adapta a las circunstancias. Por ultimo, podemos asegurar que los clientes son multiplataforma, ya podemos trabajar con diversos sistemas operativos, Linux, Solaris, OpenBSD, Windows 2000/Xp/Server , y Mac OS X.

El objetivo de este apartado es el demostrar la capacidad de conectarnos mediante un cliente VPN, en este caso nuestro dispositivo Android, al servidor VPN que se va a implementar. Para la implementación de dicho servidor se va a utilizar OpenVPN Access Server y se va a instalar en una máquina virtual con Ubuntu.

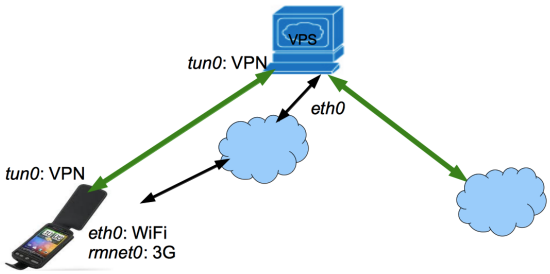

Figura 4.3: Diagrama del esquema VPN

En primer lugar, descargamos el software y lo instalamos mediante los comandos:

```
wget http://swupdate.openVPN.org/as/openVPN-as-1.8.5-Ubuntu12.
    i386.deb

sudo dpkg -i openVPN-as-1.8.5-Ubuntu12.i386.deb
```

En este punto, ya tenemos instalado OpenVPN AS. Durante dicha instalación, se ha creado un administrador por defecto llamado openVPN. A continuación, se debe aplicar una password para dicho usuario mediante el comando:

```
sudo passwd openVPN
```

En este punto, ya podemos acceder a nuestro servidor VPN mediante la url: https://direccion-ip:943/admin/ . A través de esta dirección podemos acceder a la consola de administración y configurarla como se desee (ver Fig.4.4). En nuestro caso particular, se ha tenido que cambiar la dirección IP del servidor, estableciendo la IP pública, de modo que éste fuera accesible desde otra red externa. A su vez, se ha habilitado al servidor para que sea capaz de escuchar peticiones desde todas las direcciones IP. De esta forma, la configuración del servidor queda de la siguiente manera:

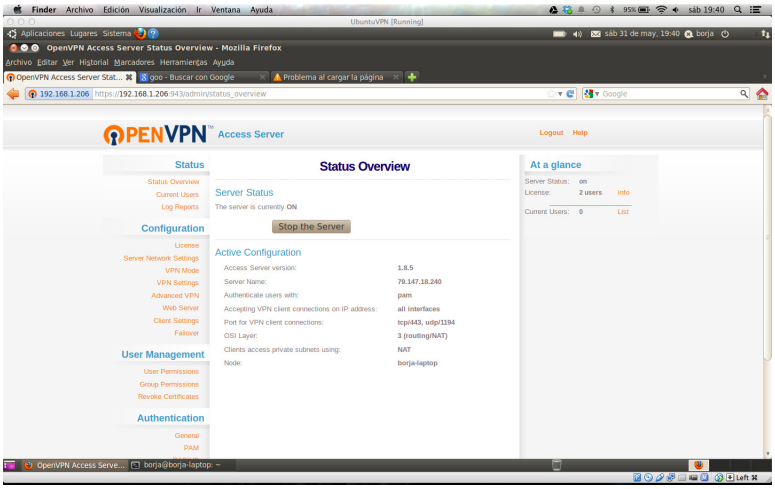

Figura 4.4: Configuración del servidor VPN

Por lo tanto, ya se tiene configurado el servidor de forma correcta. Ahora bien, es lógico que se quiera que usuarios externos puedan acceder a nuestro servidor. Para ello, se deben crear los usuarios a los cuales se les permite dicho acceso. Para ello, introducimos los siguientes comandos:

```
sudo useradd prueba

sudo passwd prueba
```

Ahora, se debe dar permiso a este nuevo usuario desde el servidor VPN. Para ello, dentro del panel de configuración, se debe acceder al apartado "User Management" y crear el nuevo usuario, de tal manera que aparezca configurado de la siguiente forma:

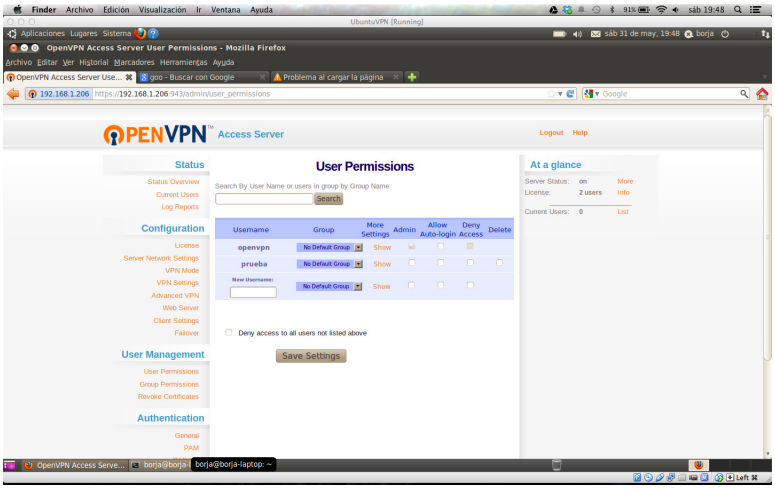

Figura 4.5: Configuración de los usuarios VPN

Una vez se haya añadido el usuario correctamente, actualizamos el servidor haciendo click en el botón "Update Running Server".

En este punto, ya se encuentra todo configurado correctamente del lado del servidor. Ahora se debe realizar la correcta configuración del cliente. Para ello, se debe obtener un fichero de configuración generado para el servidor. Nos dirigimos a la url: https://direccion-ip:943/ y nos logueamos con el usuario creado anteriormente. Descargamos el fichero de

configuración haciendo click en el botón "Yourself (user-locked profile)". Ahora ya tenemos todo lo necesario para poder acceder al servidor VPN mediante nuestro dispositivo móvil.

Por último, es necesario descargar el cliente de OpenVPN para el dispositivo móvil. En Android, dicho cliente se llama "OpenVPN Connect". Una vez descargado, únicamente es necesario adjuntar el fichero de configuración creado y conectarse al servidor. Por lo tanto, ya tenemos nuestro dispositivo móvil conectado a Internet mediante el servidor OpenVPN creado anteriormente. De esta forma, se puede navegar de formas segura en redes públicas como las de un aeropuerto, un restaurante, etc.

4.2.3. Aplicar políticas de firewall

Un *firewall* es una combinación de hardware y software que aísla la red interna de la organización de Internet, permitiendo pasar a algunos paquetes y bloqueando a otros. Un *firewall*, permite a un administrador de red controlar el acceso entre el mundo exterior y los recursos de la red administrada, gestionando el flujo de tráfico hacia y desde esos recursos (ver Fig.4.6). Un *firewall* tiene tres objetivos:

- **Todo el tráfico que va del exterior hacia el interior de la red, y viceversa, pasa a través del cortafuegos**.
- **Solo se permite el paso del tráfico autorizado de acuerdo con la política de seguridad local**. Con todo el tráfico de entrada y de salida de la red corporativa pasando a través del cortafuegos, éste puede restringir el acceso al tráfico no autorizado.
- **El propio *firewall* es inmune a la penetración**. El propio *firewall* es un dispositivo conectado a la red. Si no está diseñado o instalado apropiadamente puede verse comprometido, en cuyo caso sólo se proporciona una falsa sensación de seguridad.

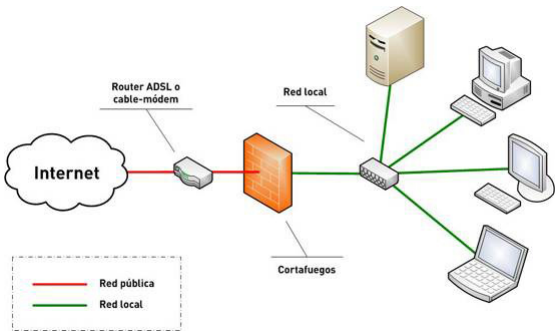

Figura 4.6: Esquema genérico de firewall

IPTables

Iptables es un *framework* disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Su finalidad principal es la de actuar como *firewall* [16]. El kernel de Linux tiene la capacidad de hacer pasar los paquetes por una serie reglas agrupadas por cadenas, que a su vez se agrupan por tablas. Al entrar un paquete pasa por todas las reglas y tablas configuradas: si existe una coincidencia se acepta "ACCEPT" o se rechaza "DROP"; si no coincide con ninguna regla se aplica la configuración por defecto.

Esta configuración por defecto se puede configurar de acuerdo a una de las siguientes estrategias:

- **Permiso predeterminado**. En este caso, cualquier paquete que no sea acorde a las reglas definidas es recibido en el sistema. Esta configuración es más sencilla de implementar pero más insegura.
- **Denegación predeterminada**. En este caso, únicamente si el paquete cumple las reglas de filtrado se acepta. En caso contrario, se rechaza. Esta configuración es más compleja de implementar, pero más segura.

Por defecto, existen cuatro tablas. A su vez, cada tabla está formada por cadenas predefinidas o definidas por el usuario. A continuación se definen las cuatro tablas principales, así como las cadenas predefinidas de cada una de ellas:

1. **Filter**. Es la tabla por defecto. Se encarga de filtrar los paquetes por la red.
   a) INPUT: paquetes destinados a un proceso local.
   b) OUTPUT: paquetes generados localmente por un proceso.
   c) FORWARD: paquetes recibidos por un dispositivo de red y que deben ser reenviados a la red sin ser procesados de forma local.
2. **NAT**. Se encarga de alterar las direcciones de origen y/o destino de los paquetes.
   a) PREROUTING: paquetes recibidos y no procesados.
   b) OUTPUT: paquetes generados localmente y no enviados.
   c) POSTROUTING: paquetes que no han salido a la red.
3. **Mangle**. Se encarga de realizar alteraciones locales del origen o destino de los paquetes para balancear tráfico, por ejemplo.
   a) PREROUTING: paquetes recibidos y no enrulados.
   b) INPUT: paquetes destinados a un proceso local.
   c) OUTPUT: paquetes generados por un proceso local y no enrutados.
   d) FORWARD: paquetes reenviados entre dos dispositivos de red.
   e) POSTROUTING: paquetes que no han salido a la red.
4. **Raw**. Se encarga de configurar excepciones en el seguimiento de los paquetes de las conexiones.
   a) PREROUTING: paquetes recibidos por cualquier dispositivo de red.
   b) OUTPUT: paquetes generados por un proceso local.

Por último, cabe destacar que IPTable tiene cuatro acciones por defecto:

- **ACCEPT**. Aceptar un paquete sin ser analizado por el resto de reglas y tablas.
- **DROP**. Rechazar un paquete sin enviar ningún mensaje al origen.
- **QUEUE**. Enviar el paquete a un módulo de procesamiento en el espacio de usuario.
- **RETURN**. Devuelve el paquete a la regla posterior a la regla que incluye la acción RETURN y que ha generado la alerta.

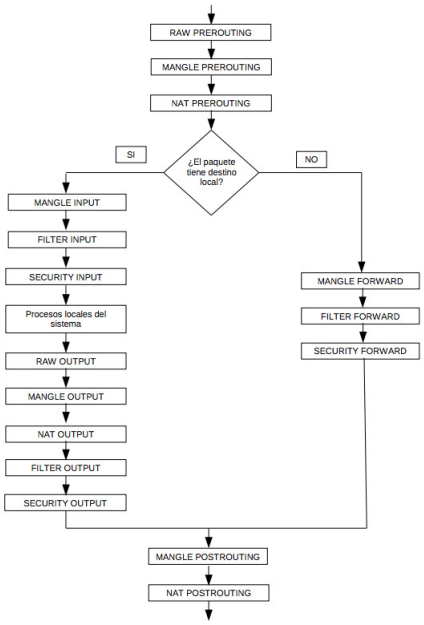

Figura 4.7: Flujo de datos de IPTables

IPChains

Ipchains es otro *framework* disponible en el kernel de Linux. Ipchains se usa para configurar, mantener e inspeccionar las reglas de cortafuegos IP del núcleo de Linux. Estas reglas se pueden dividir en cuatro categorías diferentes:

- **Cadena de entrada IP**.
- **Cadena de Salida IP**.
- **Cadena de reenvío IP**.
- **Cadenas definidas por el usuario**.

Para cada una de estas categorías se mantiene una tabla de reglas separada, cualquiera de las cuales podría remitir a una de las cadenas definidas por el usuario.

Esta herramienta permite, entre otras cosas, gestionar la entrada y salida de paquetes de los protocolos TCP, UDP e ICMP. En resumen, se le indica qué paquetes debe permitir entrar, pudiendo especificar si proceden de una IP o grupo de IPs concretos, a un puerto concreto, con un protocolo concreto, y todas las combinaciones de opciones que se decida definir. Análogamente, se definirán paquetes salientes.

IPChains consta de tres cadenas principales:

- **Input**. Aplica las reglas a los paquetes entrantes.
- **Output**. Coincide con los paquetes salientes.
- **Forward**. Coincide con los paquetes que ni proceden ni van destinados a la máquina en cuestión.

Al llegar un paquete, se analizan las reglas de la cadena input y se realiza el proceso correspondiente. Tras ello, si el paquete va dirigido a otra máquina y la máquina simplemente lo enruta, se analizan las reglas de la cadena forward y se toma la decisión que allí se indique. Posteriormente, para todos los paquetes que salen de la máquina se analizan las reglas de la cadena output y se toman acciones en consecuencia. (ver Fig.4.8)

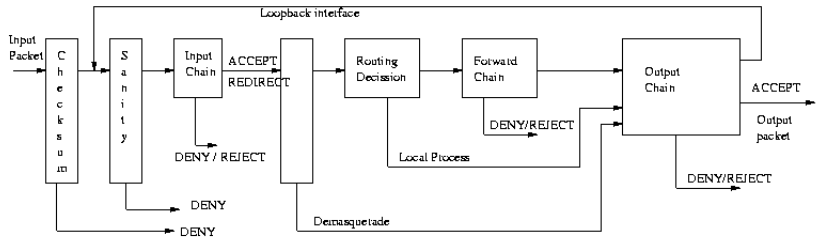

Figura 4.8: Flujo de datos de IPChains

Comparativa entre IPChains e IPTables

A primera vista, IPChains e IPTables parecen ser bastante similares. Ambos métodos de filtrado de paquetes usan cadenas de reglas operando dentro del kernel de Linux para decidir qué hacer con los paquetes que cumplen determinadas reglas. Sin embargo, IPTables proporciona un método mucho más extensible de filtrado de paquetes, dotando al administrador un nivel de control mucho más refinado sin tener que aumentar la complejidad del sistema entero. Por esta razón, en el estudio realizado en este documento, se ha elegido IPTables como método de desarrollo para implementar un *firewall*.

Los usuarios que se encuentren cómodos con IPChains deberían tener cuidado con las siguientes diferencias significativas entre IPChains e IPTables antes de utilizar IPTables:

- Bajo IPTables, cada paquete filtrado se procesa únicamente usando las reglas de una sola cadena, en lugar de hacerse con múltiples. Por ejemplo: un paquete FORWARD que llega al sistema usando IPChains tendrá que pasar por las cadenas INPUT, FORWARD, y OUTPUT para llegar a su destino. Sin embargo, IPTables sólo envía paquetes a la cadena INPUT si su destino es el sistema local y tan sólo los envía a la cadena OUTPUT si el sistema local es quien genera los paquetes. Por esta razón, es importante que se destine la regla designada para capturar un paquete particular dentro de la regla que realmente maneja el paquete.
- El objetivo DENY ha sido sustituido por DROP. En IPChains, los paquetes que coincidan con una regla en una cadena podrían ser dirigidos al objetivo DENY. Este objetivo debe ser sustituido por DROP bajo IPTables.
- El orden es importante cuando se estén definiendo opciones en una regla. En IPChains el orden de las opciones de una regla no es importante. IPTables usa una sintaxis más estricta. En comandos IPTables, el protocolo (ICMP, TCP o UDP) debe ser especificado antes del puerto fuente o destino.
- Cuando especificamos las interfaces de red que vamos a usar en una regla, deberemos utilizar sólo interfaces de entrada con cadenas INPUT o FORWARD y las de salida (opción -o) con cadenas FORWARD o OUTPUT. Esto es necesario debido a que las cadenas OUTPUT no se utilizan más con las interfaces de entrada, y las cadenas INPUT no son vistas por los paquetes que se mueven hacia las interfaces de salida.

Implementación de un firewall basado en IPTables: servidor

Como se ha indicado en el punto anterior, para este documento, se ha optado por desarrollar un *firewall* basado en IPTables.

El *firewall* se va a implementar en el mismo servidor en el que se encuentra el servidor VPN. Con dicho *firewall*, el administrador de la red será capaz de bloquear los paquetes a los usuarios que le sea conveniente.

Llevado a la práctica, lo que se pretende conseguir es que cuando un usuario esté conectado en su dispositivo móvil a la red empresarial, a través de un túnel VPN, se le bloquee el acceso a diversas páginas en base a la política empresarial de seguridad adoptada.

Para alcanzar este objetivo, se ha desarrollado el siguiente script:

```
#!/bin/sh
/sbin/iptables -F
# Tráfico de/hacia local\emph{host} y el VPN aceptado
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -o lo -j ACCEPT
/sbin/iptables -A INPUT -i tun0 -j ACCEPT
/sbin/iptables -A OUTPUT -o tun0 -j ACCEPT
# Política por defecto.
/sbin/iptables --policy INPUT DROP
/sbin/iptables --policy OUTPUT ACCEPT
/sbin/iptables --policy FORWARD ACCEPT
# Tráfico aceptado hacia los puertos 80 y 443
/sbin/iptables -A INPUT  -i eth0 -p 6 --dport 80 -j ACCEPT
/sbin/iptables -A INPUT  -i eth0 -p 6 --dport 443 -j ACCEPT
/sbin/iptables -A INPUT -p ALL -i eth0 -m state --state
   ESTABLISHED,RELATED -j ACCEPT
# NAT / Forwarding para el tráfico del VPN a Internet
/sbin/iptables -A FORWARD -i tun0 -j ACCEPT
/sbin/iptables -A FORWARD -i tun0 -s 0.0.0.0/0.0.0.0 -d
   0.0.0.0/0.0.0.0 -j ACCEPT
/sbin/iptables -A FORWARD -i eth0 -o tun0 -m state --state
   ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A FORWARD -i tun0 -o eth0 -j ACCEPT
/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE


#Bloqueo Marca
iptables -A FORWARD -d 193.110.128.199 -j REJECT
iptables -A OUTPUT -d 193.110.128.199 -j DROP
iptables -A INPUT -d 193.110.128.199 -j DROP
```

En resumen, con este script adoptamos una política en la que se rechazan todos los paquetes entrantes por defecto y se permite salir a todos los paquetes. A su vez, aceptamos el tráfico entrante y saliente hacia la VPN y aceptamos el trafico entrante de los puertos HTTP y HTTPS. Por último, se realiza un forwarding del tráfico VPN hacia Internet para poder acceder a las páginas que se permitan. Adicionalmente, se ha añadido una regla para comprobar que realmente con un script similar a este se pueden adoptar políticas de *firewall* muy estrictas y restringir las páginas de Internet inadecuadas desde el punto de vista de

la empresa. Para ello, se ha obtenido la dirección IP de la web www.marca.com (famoso diario deportivo) y se han bloqueado todos los paquetes hacia y desde dicha web.

Implementación de un firewall basado en IPTables: cliente

A su vez, se ha implementado un *firewall* en el cliente. Es decir, en el dispositivo móvil.

Llevado a la práctica, lo que se pretende conseguir es que cuando un usuario esté conectado en su dispositivo móvil, solamente pueda navegar si y solo sí tiene establecido un túnel VPN.

Para alcanzar este objetivo, se ha desarrollado el siguiente script:

```
#!/system/bin/sh
/system/bin/iptables -F
# Aceptamos el tráfico a lo
/system/bin/iptables -A INPUT -i lo -j ACCEPT
/system/bin/iptables -A OUTPUT -o lo -j ACCEPT
# Política por defecto
/system/bin/iptables --policy INPUT DROP
/system/bin/iptables --policy OUTPUT DROP
/system/bin/iptables --policy FORWARD DROP
# Reglas para el tráfico aceptado
/system/bin/iptables -A INPUT -m state --state ESTABLISHED,
   RELATED -j ACCEPT
/system/bin/iptables -A OUTPUT -m state --state ESTABLISHED,
   RELATED -p 6 --sport 22 -j ACCEPT
# Tráfico del tunel VPN
/system/bin/iptables -A INPUT -i tun0 -p 6 --dport 22 -j ACCEPT
/system/bin/iptables -A INPUT -i tun0 -p 1 -j ACCEPT
# Tráfico hacia el endpoint para establecer el túnel
/system/bin/iptables -A OUTPUT -d 81.43.193.184/32 -j ACCEPT
# Trafico que va por el túnel
/system/bin/iptables -A OUTPUT -o tun0 -j ACCEPT
```

En resumen, con este script adoptamos una política en la que se rechazan todos los paquetes por defecto. Únicamente se permite el tráfico a través del túnel VPN hacia la dirección IP 81.43.193.184, dirección IP pública de donde se encuentra situado el servidor VPN.

4.2.4. Deshabilitar Bluetooth

Es conocida la gran utilidad que aporta la conexión Bluetooth a nuestros dispositivos, como el facilitar las comunicaciones entre equipos móviles y fijos, ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

No obstante, no todo es positivo. Bluetooth ha sido vulnerable ante posibles amenazas desde que hizo su aparición. El primer malware que atacó este sistema fue el gusano Cabir, el primer gusano inalámbrico de la historia. Se transmitía a teléfonos móviles usando la

plataforma Symbian, cuando estos se encendían y activaban el modo visible. Sin embargo, los efectos no eran realmente dañinos; enviaba un mensaje con un archivo caribe.ss adjunto y una vez se descargaba el archivo se mostraba la palabra "Caribe" en la pantalla. No obstante, este primer malware fue solo una llamada de atención al sector.

Las amenazas de Bluetooth han evolucionado desde entonces y deben tomarse en serio [17].

El *bluejacking* es spam a través de Bluetooth. A través de esta técnica se envían mensajes (por ejemplo, una tarjeta de visita virtual) a usuarios en un radio de 10 metros; al descargarse dicha tarjeta, ésta añade el contacto en la agenda ya infectada.

El Car Whisperer es un software que permite a los atacantes capturar el audio de los coches que dispongan de un dispositivo manos libres. Este método permite al atacante escuchar las conversaciones y llamadas que quiera.

Bluebugging es más peligroso que los dos anteriores. Este ataque permite tener acceso remoto al teléfono del usuario y utilizar sus funciones: escucha de llamadas y envío de mensajes. Además, todo esto sucede sin que el dueño del teléfono se dé cuenta. Esto puede suponer una factura mayor de lo habitual, sobre todo si se ha utilizado el *Bluebugging* para realizar llamadas internacionales.

Los ataques a Bluetooth explotan las peticiones o procesos de permiso, que son la base de la conectividad Bluetooth. A pesar de las funciones de seguridad del teléfono, el único modo de prevenir un ataque de este tipo es desactivar el Bluetooth del dispositivo cuando no se esté usando.

Para ello, la manera más fácil en Android, es eliminar los permisos en el dispositivo Bluetooth. El Bluetooth es gestionado por el kernel a través de los dispositivos /dev/ttyHS0 y /dev/ttyMSM0. Dichos dispositivos tiene permisos tanto de lectura como de escritura. Si estos permisos son eliminados, el dispositivo Android no será alcanzable de ninguna forma.

A continuación, se exponen los comandos necesarios para ello:

```
chmod 000 /dev/ttyMSM0
chmod 000 /dev/ttyHS0
```

4.3. Configuración de sistema de un dispositivo móvil Android: almacenamiento de datos, instalación de aplicación y control de ingreso

4.3.1. Aplicar una política de password acorde a la compañía

Las contraseñas, de un tipo u otro, forman parte de nuestra vida cotidiana. Las utilizamos para Internet, en la alarma de casa y oficina, para sacar dinero de cajeros, y en el objeto de este proyecto : nuestro móvil. Muchas veces la contraseña es la única vía de acceso a los servicios y la única barrera de seguridad que separa los datos de un atacante.

Es por esto que es de suma importancia establecer una política de contraseñas correcta.

A continuación se exponen una serie de recomendaciones a la hora de aplicar una política de contraseñas definidas por INTECO [18] :

- Se deben utilizar al menos 8 caracteres para crear la clave.
- Se recomienda utilizar en una misma contraseña dígitos, letras y caracteres especiales.
- Es recomendable que las letras alternen aleatoriamente mayúsculas y minúsculas.
- Elegir una contraseña que pueda recordarse fácilmente.
- Las contraseñas hay que cambiarlas con una cierta regularidad.
- Utilizar signos de puntuación si el sistema lo permite.

En el entorno de Android, Google ofrece una API llamada "Android Device Management API" que facilita una serie de recursos para establecer ciertas reglas de seguridad en los dispositivos. Como ejemplo de esta API, se ha usado la aplicación publicada por Google, a modo de prueba, denominada "Device Management". Dicha aplicación permite establecer una política de seguridad en el dispositivo, la cual permite establecer la longitud de la contraseña y el tipo de caracteres empleados. A su vez, comprueba si la contraseña usada en el dispositivo cumple con los requisitos configurados anteriormente.

4.3.2. Deshabilitar la instalación de software

Uno de los problemas más importantes en Android es la instalación de software. Las aplicaciones en Android son empaquetadas en ficheros .apk, como se ha comentado anteriormente, y pueden ser instalados de diversas formas: a través de Internet mediante un repositorio como el Market o a través de una conexión USB. Para prevenir la instalación de software no acorde con la normativa de la empresa por parte de los usuarios, es esencial eliminar y deshabilitar el Android Market, la aplicación que permite instalar aplicaciones desde el repositorio de Google, y el fichero binario que maneja los paquetes desde la terminal (fichero "pm"). A su vez, para deshabilitar el acceso a la shell a través de USB es necesario desactivar el fichero "abdb".

Para eliminar el Android Market es necesario establecer los permisos del contenedor a 000; esto impedirá la posibilidad de lanzar el Market.

Si los permisos de ejecución son eliminados de los ficheros "pm" y "adbd", no será posible lanzarlos. Hay que tener cuidado con la ejecución de este paso, ya que debemos tener configurada la conexión SSH para, posteriormente, poder volver a habilitar los ficheros e instalar nuevo software.

A continuación, se exponen los comandos necesarios a ejecutar en el dispositivo, mediante la conexión SSH, para deshabilitar la instalación de software:

```
chmod -x /system/bin/pm
mount -o rw,remount -t rootfs rootfs /
chmod -x /sbin/abdb
mount -o ro,remount -t rootfs rootfs /
killall com.Android.vending
chmod 000 /data/data/com.Android.vending
```

4.3.3. Deshabilitar tarjeta SD

Conectar un dispositivo externo a un sistema seguro puede ser un gran riesgo de seguridad independientemente de si es en un *smartphone*, en un servidor o en un ordenador. En el caso de Android, la situación es la misma o incluso peor. Por defecto, Android monta la tarjeta SD como un sistema FAT con permisos "noexec" y "nodev". La primera cuestión es por qué usa FAT en lugar de ext3 o ext4 pero, aparte de esto, ¿es el permiso "noexec" suficiente para prevenir la ejecución de aplicaciones? La respuesta a esta pregunta está en manos del desarrollador. Es la figura de éste quien otorga permisos para acceder a la tarjeta SD o no. En caso de facilitar los permisos necesarios a las aplicaciones, un malware que se encuentre en el dispositivo tendría permisos de ejecución sustituyendo el fichero correcto de la apk instalada. Por lo que, desde un punto de vista empresarial, es recomendable deshabilitar el acceso a la tarjeta SD para evitar posibles ejecuciones de malware en el dispositivo.

A la hora de deshabilitar la tarjeta SD, es necesario desmontarla durante el proceso de inicio. Para deshabilitarla, es preciso ejecutar los siguientes comandos:

```
umount /mnt/sdcard/.Android_secure
umount -l /mnt/sdcard
mount -o ro,remount /system
```

4.3.4. Deshabilitar binarios innecesarios

Como parte de un proceso de mejora de la seguridad en un sistema general de información, una buena práctica consiste en eliminar los compiladores y los paquetes innecesarios.

En el caso particular de Android, hay que tener en consideración los binarios con acceso a la red, editores o sniffers. Dependiendo del caso, es necesario borrar los archivos o simplemente cambiar los permisos de estos.

A continuación, se exponen los comandos necesarios a ejecutar en el dispositivo para borrar los binarios innecesarios:

```
rm -f /system/xbin/irsii
rm -f /system/xbin/nano
rm -f /system/xbin/nc
rm -f /system/xbin/netserver
rm -f /system/xbin/netperf
rm -f /system/xbin/opcontrol
chmod 740 /system/xbin/scp
chmod 740 /system/xbin/rsync
chmod 740 /system/xbin/sdptest
chmod 740 /system/xbin/SSH\index{SSH}
chmod 740 /system/xbin/strace
chmod 000 /system/xbin/tcpdump
chmod 740 /system/xbin/vim
chmod 000 /system/bin/bluetoothd
chmod 750 /system/bin/iptables
chmod 750 /system/bin/ping
chmod -s /system/bin/ping
```

Es importante destacar que no todos los ficheros expuestos se encuentran en todas las versiones de Android. Dependiendo de la versión usada, habrá ficheros que no existan por defecto. Por ejemplo, en el dispositivo que se ha usado en las pruebas, funcionando con Android 4.1.2, el script quedaría de la siguiente forma:

```
rm -f /system/xbin/nc
chmod 750 /system/bin/iptables
chmod 750 /system/bin/ping
chmod -s /system/bin/ping
```

4.3.5. Eliminar software innecesario

La mayoría de las compañías poseen un registro con las aplicaciones aprobadas para su uso, que se aplica a las estaciones de trabajo. La idea que se persigue es la misma: definir una serie de aplicaciones estándar que se instalen y eliminar el resto.

Las aplicaciones que se entienden indispensables, y se van a permitir por defecto, son: un navegador de Internet, un cliente de correo, una calculadora, un calendario, una aplicación para la cámara, los contactos, la galería de imágenes, los mensajes y el teléfono. Si en algún caso fuera necesaria alguna otra aplicación, el administrador de seguridad podría instalarla subiendo la apk al dispositivo mediante SCP (siempre a través de VPN), cambiar los permisos del fichero "pm" a ejecutable, instalar la aplicación y revertir los cambios. Este modelo es muy flexible y robusto debido a que es seguro y cualquier aplicación puede ser instalada remotamente sin la interacción del usuario.

Para borrar el software innecesario, es necesario eliminar los apk y los directorios donde se encuentran instaladas las aplicaciones. Los paquetes de aplicación se encuentran en /system/app, mientras que los datos se encuentran en /data/data. Una buena forma de realizar este proceso para poder revertirlo en algún momento es cambiar los permisos de los directorios a 000 en lugar de eliminarlos.

A continuación, se expone un ejemplo del proceso a seguir por cada aplicación que se desee borrar. En este caso, se va a deshabilitar "Dropbox":

```
rm /system/app/Dropbox.apk
chmod 000 /data/data/com.dropbox.Android
```

4.3.6. Remote Wipe

Una de las mayores preocupaciones a nivel empresarial es la pérdida o robo del dispositivo móvil y, con ello, toda la información confidencial almacenada en él. Es por eso, que actualmente, es necesario aportar la funcionalidad del bloqueo y borrado de datos remoto.

Existen varias herramientas que permiten esta función. En este estudio se va a usar la herramienta "Device Manager" desarrollada por Google (ver Fig.4.9) [19]. Dicha herramienta permite obtener la localización del dispositivo, así como borrar todos los datos o bloquear el dispositivo. Todo ello de forma remota.

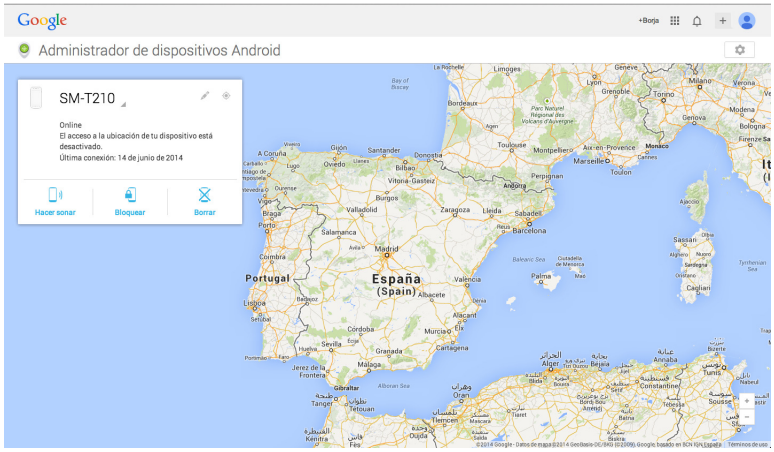

Figura 4.9: Device Manager de Google

4.3.7. Antivirus

Una capa de seguridad muy importante en una arquitectura de defensa es el antivirus. La popular compañía AVG ha desarrollado una versión de antivirus para Android que permite escanear aplicaciones y ficheros en tiempo real. A su vez, posee una opción para navegar por internet de forma segura. Existe una funcionalidad para encontrar el dispositivo en Google Maps, para hacer un *backup* o una restauración de todas las aplicaciones y para bloquear el dispositivo.

Para la realización de este estudio únicamente será utilizado el motor de AVG, ya que las demás funcionalidades se han cubierto con otras herramientas.

Para obtener la aplicación únicamente es necesario acceder desde el Android Market y buscar "AVG".

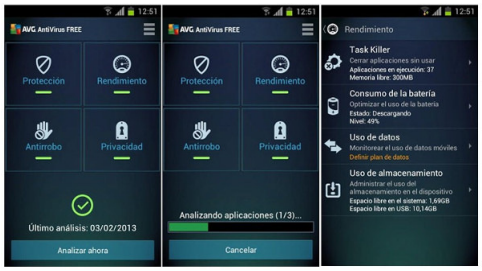

Figura 4.10: AVG

Capítulo 5

Monitor de tráfico

Con el objetivo de detectar posibles intrusiones, muchas técnicas han sido usadas a lo largo del tiempo. Estas técnicas incluyen herramientas de captura de paquetes y sistemas de detección de intrusos (IDS). Con el desarrollo de las plataformas móviles, como Android, el concepto de LAN ha cambiado sustancialmente. Actualmente, cualquier empleado puede usar su *smartphone* para conectarse a la red de su compañía y realizar tareas tan triviales como consultar el correo electrónico. Como consecuencia, hay nuevos vectores de ataque que pueden ser usados para comprometer la seguridad de las infraestructuras de la compañía.

En este punto se pretende demostrar como implementar un arquitectura que permita monitorear el tráfico del *smartphone* y detectar conexiones no autorizadas o dispositivos comprometidos.

5.1. Sistema de detección de intrusos (IDS)

Un IDS es una herramienta de seguridad que intenta detectar o monitorizar los eventos ocurridos en un determinado sistema informático en busca de intentos de comprometer la seguridad de dicho sistema. Los IDS buscan patrones previamente definidos que impliquen cualquier tipo de actividad sospechosa o maliciosa sobre nuestra red o *host*. Los IDS aportan a nuestra seguridad una capacidad de prevención y de alerta anticipada ante cualquier actividad sospechosa. No están diseñados para detener un ataque, aunque sí pueden generar ciertos tipos de respuesta ante éstos.

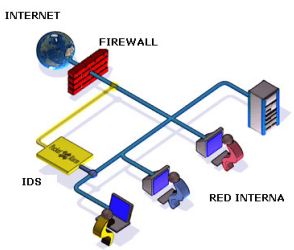

Figura 5.1: Sistema de detección de intrusos

Existen dos tipos de sistemas de detección de intrusos:

- **HIDS**. Protege contra un único servidor, PC o *host*. Monitorizan gran cantidad de eventos, analizando actividades con una gran precisión, determinando de esta manera qué procesos y usuarios se involucran en una determinada acción. Recaban información del sistema como ficheros, logs, recursos, etc, para su posterior análisis en busca de posibles incidencias. Todo ello en modo local.

- **NIDS**. Protege un sistema basado en red. Actúa sobre una red capturando y analizando paquetes de red. Es decir, son sniffers del tráfico de red. Luego analizan los paquetes capturados, buscando patrones que supongan algún tipo de ataque. Bien ubicados, pueden analizar grandes redes y su impacto en el tráfico suele ser pequeño. Analizan el trafico de red, normalmente, en tiempo real. No sólo trabajan a nivel TCP/IP, también lo pueden hacer a nivel de aplicación.

A su vez, los IDS se pueden clasificar por el tipo de respuesta otorgada:

- **Pasivos**. Son aquellos IDS que notifican a la autoridad competente o administrador de la red mediante un sistema predefinido, alerta, etc. Y no actúa sobre el ataque o atacante.

- **Activos**. Generan algún tipo de respuesta sobre el sistema atacante o fuente de ataque (como cerrar la conexión o enviar algún tipo de respuesta predefinida en nuestra configuración).

A grandes rasgos, la arquitectura de un IDS está compuesta por:

1. La fuente de recogida de datos. Estas fuentes pueden ser un log, dispositivo de red, o como en el caso de los IDS basados en *host*, el propio sistema.

2. Reglas que contienen los datos y patrones para detectar anomalías de seguridad en el sistema.

3. Filtros que comparan los datos escaneados de la red o de logs con los patrones almacenados en las reglas.

4. Detectores de eventos anormales en el tráfico de red.

5. Dispositivo generador de informes y alarmas.

Una cuestión muy importante para los administradores de sistemas es dónde colocar el IDS. Una actitud paranoica puede llevar a instalar un IDS en cada *host* o en cada tramo de red. Esto último podría tener una cierta lógica cuando se trata de grandes redes. Lo coherente sería instalar el IDS en un dispositivo por donde pase todo el tráfico de red que nos interese.

Otro aspecto a tener en cuenta es si colocar el IDS antes o después del *firewall*. Si colocamos el IDS antes del cortafuegos capturaremos todo el tráfico de entrada y salida de nuestra red. La posibilidad de falsas alarmas es grande. La colocación detrás del cortafuegos monitorizará todo el tráfico que no sea detectado y parado por el *firewall*, por lo que será considerado como malicioso en un alto porcentaje de los casos. La posibilidad de falsas alarmas es muy inferior.

Para este estudio, se va a utilizar el sistema de detección de intrusos Snort.

5.2. IDS existentes en el mercado

Los IDS existentes en el mercado se pueden dividir en dos categorías claramente diferenciadas: IDS comerciales e IDS de dominio público. A continuación, se va a realizar una revisión resumida de los IDS existentes en el mercado.

Una vez realizado el estudio de los IDS existentes, se optó por una herramienta de código libre, dado el abaratamiento de costes empresariales y a la gran comunidad existente alrededor de estas herramientas. Por ello, se eligió Snort como el IDS a utilizar, ya que proporciona utilidades más que suficientes para el alcance de este documento y posee una gran cantidad de complementos adicionales que funcionan bajo Snort, como es el caso de Snorby.

IDS comerciales

Proventia

Es un producto de IBM que permite protección preventiva para el núcleo de redes comerciales. Posee flexibilidad de puertos y una capacidad de análisis para tráficos de hasta 2 Gbps, cumpliendo satisfactoriamente con los requerimientos de las grandes empresas. Entre sus principales características se encuentran:

- Arquitectura de administración escalable de tres niveles.
- Arquitectura de control de eventos y de red centralizada.
- Análisis de seguridad y motor de base de datos centralizado.
- Análisis de datos orientado a grupos.
- Soporte para productos ISS (Information System Solution).

Cisco IPS-4255

Este producto de Cisco provee detección pasiva IDS e IPS (Sistema de Prevención de Intrusos) hasta velocidades de 500 Mbps, cumpliendo los requerimientos de las empresas medianas. Es muy estable ante ataques prolongados de DOS y DDOS. Permite la administración a través de una interfaz de linea de comandos o con una administración centralizada que permite gestionar muchos dispositivos distribuidos a lo largo de la red.

McAfee IntruShield

Este producto está basado en una combinación de procesadores estándares y personalizados. El dispositivo McAfee IntruShield es un dispositivo de alto rendimiento ofreciendo detección y prevención de intrusiones contra ataques conocidos y desconocidos en tiempo real para redes corporativas.

El sistema IntruShield es capaz de operar en redes a velocidades altas (hasta 2 Gbps), y es capaz de operar en línea o como IDS pasivo. Posee la capacidad de definir IPS virtuales,

lo cual permite definir políticas de manera independiente para diferentes niveles llegando hasta el nivel del servidor si se requiere.

Check Point IPS-1

Check Point IPS 1 es un sistema detector de intrusiones dedicado que ayuda a las organizaciones a asegurar sus redes corporativas, y proteger a los servidores críticos contra gusanos, malware automáticos y ataques combinados conocidos y desconocidos.

IPS 1 provee una seguridad fuerte robusta y dinámica. Además IPS-1 provee herramientas administrativas que incrementan la eficacia de los administradores. Asimismo, la administración de este producto se encuentra integrada con el resto de los productos de Check Point, reduciendo costes de formación e incrementando la eficiencia de los operadores.

IDS de dominio público

Shadow

El proyecto Shadow es un desarrollo conjunto de Centro de Armas Navales "Dahlgren", de la Agencia Nacional de seguridad (NSA) y del instituto SANS. Shadow usa estaciones de censado y de análisis. Los sensores usualmente se hallan ubicados en puntos claves de la red, tales como puntos fuera del *firewall*, mientras que las estaciones de análisis se encuentran en la zona interna del *firewall*. Los sensores están basados en programas de dominio publico de captura de paquetes y estos no procesan los datos, de esta forma evita que un intruso determine los objetivos de detección capturando un sensor. Los sensores extraen el encabezamiento de los paquetes y los almacenan en un archivo que la estación de análisis lee periódicamente. El principal soporte es de tcpdump.

La estación de análisis usa una interfaz basada en web para mostrar los resultados. Shadow se puede ejecutar en cualquier sistema Unix incluido FreeBSD y Linux.

Blare

Blare es un detector de intrusiones basado en políticas, que corre sobre plataforma Linux. Su principal propósito es servir como banco de pruebas para experimentar con nuevos sistemas de detección de intrusiones. A diferencia de otros sistemas de detecciones como Snort, Blare no requieres firmas de ataques o perfiles de aprendizaje, ni conocimiento de los programas atacantes. Sus principales objetivos son:

- Detectar todas las violaciones de las políticas de seguridad, incluyendo violaciones mediante ataques nuevos y desconocidos.
- Reportar solo las violaciones de las actuales políticas de seguridad, sin falsos positivos.
- Permite trabajar con políticas usuales de seguridad, tales como el acceso discrecional

Snort

Snort es un IDS basado en red (NIDS). Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida como patrones que corresponden a ataques, barridos, intentos de comprometer alguna vulnerabilidad, análisis de protocolos, etc. Todo esto en tiempo real [20].

Snort está disponible bajo licencia GPL, gratuito y funciona bajo plataformas Windows y UNIX/Linux. Es uno de los más usados y dispone de una gran cantidad de filtros o patrones ya predefinidos, así como actualizaciones constantes ante casos de ataques, barridos o vulnerabilidades que vayan siendo detectadas, a través de los distintos boletines de seguridad.

Este IDS implementa un lenguaje de creación de reglas flexibles, potente y sencillo. Durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, ddos, finger, ftp, ataques web, CGI, escaneos Nmap, etc.

Puede funcionar como sniffer, como registrador de paquetes o como un IDS normal (en este caso NIDS).

Una característica muy importante e implementada desde hace pocas versiones es FlexResp. Permite, dada una conexión con tráfico malicioso, darla de baja. Esto es: hacerle un DROP mediante el agregado de una regla en el *firewall*, de tal forma que envié un paquete con la bandera RST activa, con lo cual cumpliría funciones de IPS (Intrusion Prevention System), cortando las conexiones que cumplan ciertas reglas predefinidas. No sólo corta la conexiones, puede realizar otras acciones adicionales.

5.3. Instalación y configuración de Snort

Para instalar Snort en Ubuntu, es necesario ejecutar:

```
sudo apt-get install snort.
```

Una vez instalado Snort, es necesario implementar las reglas. Para ello, se han instalado las reglas de "Emerging Threats". Estas reglas han de ser copiadas en la carpeta /etc/snort/rules. En concreto, se van a utilizar las reglas: emergingmobile_malware.rules, emerging-web_client.rules, emergingtrojan.rules, emergingmalware.rules y emergingbotcc.rules . Las reglas que se definan por parte del usuario han de ser almacenadas en el fichero local.rules de la misma carpeta [21].

El siguiente paso consiste en definir la red y la interfaz que se quiere monitorizar. Para ello, es necesario modificar el fichero /etc/snort/snort.debian.conf donde la red y la interfaz están definidos de la siguiente forma:

```
DEBIAN_SNORT_HOME_NET=''192.168.1.1/24''
DEBIAN_SNORT_INTERFACE=''tun0''
```

Por último, es necesario iniciar Snort mediante el comando:

```
sudo /etc/init.d/snort start
```

Para comprobar el correcto funcionamiento de Snort se puede ejecutar el siguiente comando:

```
sudo snort -c /etc/snort/snort.conf -T
```

5.3.1. Capturando el tráfico

A la vez que se va a monitorizar todo el tráfico mediante Snort, se va a capturar éste mediante "tcpdump". Tcpdump permite al usuario capturar y mostrar a tiempo real los paquetes transmitidos y recibidos en la red a la cual el ordenador está conectado.

Para lanzar tcpdump y capturar todo el tráfico es necesario ejecutar el siguiente comando:

```
sudo tcpdump -i eth0 -s 0 -w /home/borja/captura.cap
```

Una vez tengamos el fichero .cap podemos analizarlo mediante "Wireshark" o mediante tcpdump. Para analizarlo mediante tcpdump es necesario ejecutar el siguiente comando:

```
sudo tcpdump -nr /home/borja/captura.cap
```

5.3.2. Creación de reglas propias y testeo de ellas

De una forma esquemática, este es el aspecto y la estructura que tiene una regla de Snort:

acción protocolo_a_vigilar ip_origen puerto_origen -> ip_destino puerto_destino [referencia: valor; ...]

Dichos campos corresponden con los siguientes puntos:

- Acción a realizar.
- Protocolos.
- Direcciones IP.
- Puertos.
- Operador de dirección.
- Opciones de las reglas.

Acción a realizar

El primer campo que se encontrará en toda regla, determinará la acción a realizar con los paquetes en caso de que cumpla las condiciones configuradas en el filtro. Puede tomar uno de los siguientes valores:

alert	Genera una alerta usando el método elegido y luego registra el paquete en un log
log	Registra el paquete en un log
pass	Ignorar el paquete
activate	Genera una alerta y luego activa otra regla dinámica
dynamic	Permanece desactivada hasta que se activa con una regla ”activate”, luego actua como una regla ”log”
drop	Bloquea y registra el paquete en un fichero log.
reject	Bloquea el paquete, lo registra y luego envía un TCP reset si el protocolo es TCP o un ”ICMP port unreachable” si el protocolo es UDP.
sdrop	Bloquea el paquete pero no lo registra en un log

Snort nos ofrece la posibilidad de crear tipos de reglas personalizados, y usarlos posteriormente como acciones en Snort, pero no vamos a entrar en detalles al respecto. Más adelante, se estudiará la creación de reglas propias.

Protocolos

El siguiente campo en una regla es el protocolo, que debe ser uno de los cuatro que Snort puede analizar actualmente: TCP, UDP, ICMP o IP.

Direcciones IP y puerto

El siguiente campo hace referencia a la dirección IP y el puerto de origen; tras el operador de dirección también se especifican la dirección IP y el puerto de destino.

Operador de dirección

Indica la orientación o dirección del tráfico sobre el que se aplicará la regla. Puede ser ”>” o ”<>” (bidireccional).

Opciones

A partir de aquí podemos añadir las opciones que nos interesen, separadas por ”;”. Existen cuatro tipos de categorías de opciones: general, payload, non-payload y post-detection.

Opciones de tipo general

Dan información sobre la regla pero no tienen efecto alguno a la hora de la detección. A continuación se presenta una tabla con dichas opciones y sus funcionalidades.

msg	El mensaje que se mostrará en el tipo de alerta seleccionada cuando un paquete coincida con la descripción de la regla.
reference	Permite incluir referencias a sistemas de identificación de vulnerabilidades externos.
GID	Se usa para identificar qué parte de Snort genera el evento cuando una regla se ”dispara”
sid	Se usa para identificar reglas.
rev	Se usa para identificar revisiones de las reglas.
classtype	Sirve para categorizar una regla por la forma en la que se detecta un ataque que pertenece a otro tipo de ataque más general.
priority	Asigna un nivel de importancia a las reglas.
metadata	Permite añadir información adicional sobre la regla.

Opciones de detección de payload

Estas opciones buscan datos dentro de los paquetes. Debido a la cantidad de opciones disponibles, se presentan las más habituales en formato de tabla-resumen.

content	Permite al usuario definir reglas que busquen un contenido específico en el paquete y active una respuesta basado en ese contenido.
rawbytes	Permite a las reglas mirar el contenido del paquete en ”raw”.
depth	Define hasta donde debe buscar Snort dentro de un paquete.
offset	Especifica donde debe comenzar a buscar un patrón en los paquetes.

Opciones de detección de non-payload

Con estas opciones podemos hacer que las reglas busquen por datos que no sean payloads. En total son veinte las opciones non-payload disponibles, por lo que se muestra una tabla con las más relevantes.

ttl	Comprueba el valor time-to-live.
dsize	Comprueba el tamaño del paquete payload.
flags	Comprueba si bits específicos de las flags TCP están presentes.
seq	Busca un número de secuencia TCP en concreto.
ack	Busca un ”acknowledge number” en concreto.
window	Busca un tamaño de ventana TCP en concreto.
icmp_id	Busca un valor específico de ICMP ID.
sameip	Permite que las reglas comprueben si la IP de origen es la misma que la IP de destino.

Opciones de post-detection

Tenemos a nuestra disposición varias opciones de post-detección que se pondrán en marcha cuando una regla se active. Con estas opciones podremos terminar de adaptar las

reglas a las necesidades que tengamos.

logto	Registra todos los paquetes que activen la regla en un fichero determinado.
session	Extrae información del usuario de las sesiones TCP.
resp	Intenta cerrar sesiones cuando una alerta se activa.
react	Permite a los usuarios terminar la conexión y enviar un aviso.
tag	Permite registrar más que solo el paquete que activó la regla.
activates	Permite especificar una regla que se añadirá cuando ocurra un evento en la red.
activated_by	Permite activar dinámicamente una regla.
count	Se debe usar junto a "activated_by". Permite especificar durante cuantos paquetes estará la regla activa tras su activación.
detection_filter	Busca por dirección IP otras reglas que coincidan y las activa.

Como se puede observar, Snort ofrece un gran conjunto de herramientas para personalizar las reglas de la forma deseada. En el contexto empresarial en el que se centra este estudio, las reglas han de definirse tomando como punto de partida la política de seguridad de la empresa. A su vez, dichas reglas han de ajustarse a cambios de la política de seguridad, bien provocados por la detección de nuevas amenzas, por configuraciones que impiden un uso eficiente de las tecnologías corporativas, o simplemente por otro motivo justificado. Por lo tanto, la configuración y adaptación de dichas reglas son tareas que debe llevar a cabo el administrador de sistemas de la empresa, o bien el responsable de seguridad.

A modo de ejemplo, se va a crear una regla personalizada y se va a comprobar si la alerta se dispara adecuadamente. En dicha regla, se van a definir los siguientes puertos como permitidos: 53/tcp, 53/udp, 80/tcp, 123/udp, 443/tcp y 5228/tcp. Estos puertos se corresponden a DNS, HTTP/S y al tráfico de Google Play.

Una vez definida la regla, hay que configurarla. Cabe destacar que la alerta de la regla se debe disparar cuando se reciba tráfico en otro puerto de los que no se han permitido. La definición de la regla quedaría de la siguiente forma:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET !$HTTP_PORTS,!5228
,!53 (msg: ''Puertos TCP no estándar''; sid:99999999; rev:0;)

alert udp $HOME_NET any ->$EXTERNAL_NET !123,!53
 (msg: ''Puertos UDP no estándar''; sid:99999999; rev:0;)
```

Para testear la regla, se va a establecer una conexión desde un dispositivo Android usando la herramienta netcat. Se va a especificar una dirección IP aleatoria y un puerto de destino igual a 9999. Después se verificará que se ha generado la alerta en el fichero /var/log/snort/alert. Hay que reseñar que dicha conexión se debe establecer siempre a través de la conexión VPN. El fichero de alerta generado es el siguiente:

[**] [1:99999999:0] PUERTOS TCP NO ESTANDAR [**] [Priority: 0] 06/15-19:37:21.311592 192.168.1.206:37049 -¿63.245.216.132:443 TCP TTL:64 TOS:0x0 ID:23800 IpLen:20 DgmLen:40 DF ***A***F Seq: 0xF4856469 Ack: 0xF91E2479 Win: 0x3EA TcpLen: 20

Por otro lado, se debe comprobar si la alerta ha sido lanzada adecuadamente. Para ello, hay que analizar la traza de tráfico capturada por tcpdump. Si se estudia correctamente, se puede encontrar la siguiente sentencia:

19:06:47.664402 IP 193.152.169.173.9999 ¿192.168.1.206.48886: Flags [R.], seq 0, ack 1191139047, win 0, length 0

Con la cual se puede asegurar que la alerta lanzada por Snort ha sido disparada correctamente ya que tcpdump indica que se ha hecho una petición al puerto 9999, el cual no estaba habilitado.

Como se ha comentado anteriormente, de forma análoga se pueden configurar todo tipo de reglas para detectar conexiones no deseadas en la red corporativa.

Capítulo 6

Administración y gestión centralizada de dispositivos Android: sistemas MDM

Con objeto de tener un producto integral y una interfaz cómoda para el administrador se han desarrollado los administradores de dispositivos (Mobile Device Management).

Se conoce por Mobile Device Management (MDM) a aquel software que permite asegurar, monitorear y administrar dispositivos móviles sin importar el operador de telefonía o proveedor de servicios. La mayoría de los MDM permiten hacer instalación de aplicaciones, localización y rastreo de equipos, sincronización de archivos, reportes de datos y acceso a dispositivos, todo de manera remota. Este tipo de software ha tenido una gran aceptación por parte de las empresas y su crecimiento ha sido realmente vertiginoso, debido en gran parte a la popularidad que han alcanzado los *smartphones* en las corporaciones.

La arquitectura básica de un MDM consiste en un agente, el cual es una aplicación que se instala en cada uno de los dispositivos que se desean administrar, un servidor de implementación desde donde se ejecuta el MDM y una base de datos donde se guardan todos los datos recabados (ver Fig.6.1). Los agentes mantienen una conexión con el servidor a través de USB, Wi-Fi, GPRS, 3G o cualquier otro medio de transmisión de datos lo cual le permite al MDM tomar control del dispositivo.

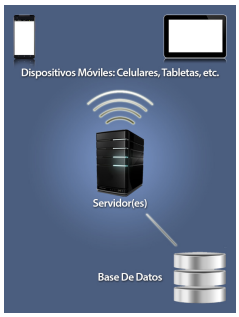

Figura 6.1: Arquitectura de un MDM

6.1. MDM existentes en el mercado

AirWatch

La administración de dispositivos móviles de AirWatch permite a las empresas enfrentar los retos asociados a la movilidad al proveer una manera simple y eficiente de ver y administrar todos los dispositivos desde una sola consola de administración central. La solución permite registrar rápidamente dispositivos en su entorno empresarial, configurarlos, actualizar los ajustes de los dispositivos over-the-air y securizarlos.

Sus principales características son:

- Consola de administración única.
- Sencillo proceso de registro.
- Perfiles que permiten definir configuraciones, políticas y restricciones empresariales para los dispositivos sin requerir la interacción de los usuarios.

SAP Afaria

SAP Afaria es un programa de gestión de movilidad empresarial que permite:

- Proteger los datos corporativos importantes con una arquitectura segura y ampliable
- Elegir métodos de implementación flexibles y ampliables, ya sea en la nube o en local.
- Respaldar escenarios de dispositivos personales como corporativos (Bring Your Own Device).

OwnMDM

El software que se va a utilizar para este estudio es "ownMDM". ownMDM es un gestor de dispositivos, que permite controlar remotamente los dispositivos registrados desde un servidor.

Es un software libre, y consta de dos módulos:

- **Web**. La parte web es un php que usa una base de datos mySql para enviar y recibir los datos a los dispositivos registrados.
- **App**. La app es un administrador móvil que funciona como cualquier otro, haciendo las tareas de forma transparente.

A continuación se van a exponer los pasos a seguir para la correcta instalación de la parte web de ownMDM:

```
# Actualizacion e instalación de las librerías de Apache
sudo apt-get update
sudo apt-get install apache2 php5 libapache2-mod-php5

# Se inicia el servidor Apache
service apache2 start

# Se instala Curl y MySql
sudo apt-get install php5-curl
sudo apt-get install mysql-server mysql-client php5-mysql

# Se reinicia Apache
sudo service apache2 restart

# Se instala phpmyadmin. Aunque no es imprescindible facilita el
    manejo de la base de datos
sudo apt-get install phpmyadmin
```

Para terminar de configurar phpmyadmin, es necesario añadir al fichero /etc/apache2/apache2.conf la sentencia:

```
Include /etc/phpmyadmin/apache.conf
```

Una vez realizado este paso, ya podemos iniciar phpmyadmin desde el navegador, introduciendo "http://ip_pc/phpmyadmin".

Una vez se encuentran instaladas las librerías necesarias, se creará la base de datos donde se registrarán todas las acciones del MDM. Para ello, es necesario ejecutar el siguiente script:

```
SET SQL_MODE='NO_AUTO_VALUE_ON_ZERO';
CREATE DATABASE `MDM\index{MDM}` DEFAULT CHARACTER SET latin1
    COLLATE latin1_swedish_ci;
USE `MDM\index{MDM}`;

-- --------------------------------------------------------


--
-- Estructura de la table 'devices'
--

CREATE TABLE IF NOT EXISTS `devices` (
  `imei` varchar(100) NOT NULL,
  `gcm` varchar(300) default NULL,
  `location` varchar(200) default NULL,
  `dateLocation` timestamp NULL default NULL,
  `enabled` tinyint(4) default NULL,
  `model` varchar(300) default NULL,
  `enabledDate` timestamp NULL default NULL,
  `disabledDate` timestamp NULL default NULL,
  `ping` timestamp NULL default NULL,
  PRIMARY KEY (`imei`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--
-- Estructura de la tabla 'devices_log'
--

CREATE TABLE IF NOT EXISTS `devices_log` (
  `imei` varchar(100) NOT NULL,
  `log` varchar(500) NOT NULL,
  `dateLog` timestamp NOT NULL default CURRENT_TIMESTAMP on
     update CURRENT_TIMESTAMP,
  KEY `imei` (`imei`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

ALTER TABLE `devices` ADD `name` VARCHAR( 200 ) NULL AFTER `imei
    `
```

Por último es necesario crear el usuario "MDM" con una contraseña y darle privilegios. Para ello, es necesario ejecutar:

```
CREATE USER 'MDM\index{MDM}'@'%' IDENTIFIED BY '***';

GRANT ALL PRIVILEGES ON *.* TO 'MDM\index{MDM}'@'%' IDENTIFIED
    BY '***' WITH GRANT OPTION
  MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
     MAX_UPDATES_PER_HOUR 0
  MAX_USER_CONNECTIONS 0 ;

GRANT ALL PRIVILEGES ON `MDM\index{MDM}`.* TO 'MDM\index{MDM}'
    @'%';
```

Una vez realizado el proceso de configuración, es hora de instalar el MDM. Es necesario descargarse el código de la parte web desde: https://github.com/pacosal/ownMDMPortal.

Cuando esté descargado, copiamos todos los ficheros a la carpeta /var/www, que es donde se encuentran los ficheros accesibles por Apache. A su vez, es necesario cambiar los permisos a la carpeta MDM. Posteriormente, debemos editar el fichero /var/www/MDM/web/init.inc.php cambiando el valor de mykey por la clave que se desee aplicar, que deberá ser la misma que la introducida en la app del dispositivo móvil. También se puede introducir el correo electrónico para recibir fotos y audios.

Finalmente, ya estaría todo configurado en la parte del cliente. Se puede acceder a la interfaz web mediante "http://ip_pc/MDM/index.php".

La parte del dispositivo móvil es mucho más sencilla. Únicamente es necesario descargarse la app desde Google Play (https://play.google.com/store/apps/details?id=com.pacosal.MDM) y configurarla. La configuración consiste en introducir la dirección IP y el puerto de la máquina donde reside la parte web. Una vez realizada, podemos observar que en la parte web se ha añadido el dispositivo:

Las funcionalidad que ofrece este MDM son las siguientes:

- **Message**. Manda una notificación configurada por el usuario al dispositivo.

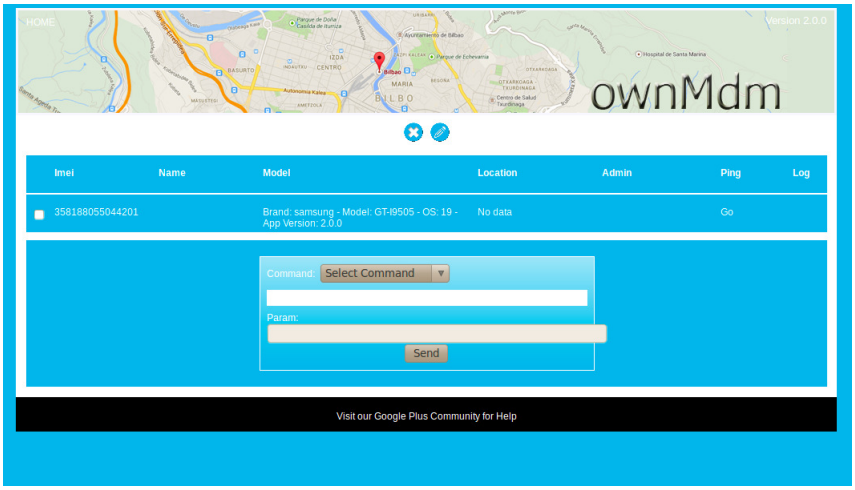

Figura 6.2: Ejemplo de ownMDM

- **Lock**. Bloquea la pantalla del dispositivo.
- **Ring**. Emite un sonido tipo sirena de policía.
- **Enable Admin**. El dispositivo recibe una notificación para que active la aplicación si no está activa.
- **Location**. El dispositivo responde con su localización de Google Maps y sus redes WiFi.
- **Location Alarm**. Envía una alerta cuando el dispositivo abandona la localización actual.
- **Wipe**. El dispositivo es completamente borrado.
- **Lock with Key**. Bloquea el dispositivo con una contraseña configurada por el administrador.
- **Force update Model**. Obtiene el modelo de datos.
- **Record Audio**. Graba un audio durante 20 segundos y lo manda al email configurado.
- **Take a Picture**. El dispositivo toma una foto y la manda al email.
- **Take a Video**. El dispositivo graba un video y lo manda al email.
- **Receive a SMS**. Ayuda a obtener el número móvil y el IMEI insertado en el dispositivo.
- **Track Device**. El dispositivo devolverá su localización, el sonido e imágenes cada 3 minutos durante un período. de 15 minutos.
- **Receive a File**. El dispositivo mandará el fichero introducido por parámetro.

Capítulo 7

Líneas futuras

Como se ha podido observar a lo largo de este documento, se ha trabajado con la motivación de que la labor realizada en este contexto no se limite como fin en sí misma sino que deje abiertas distintas posibilidades de investigación y futuro trabajo.

La línea de futuro más clara es la integración de todas las herramientas cubiertas bajo una misma aplicación, en la cual poder gestionar y configurar las políticas de seguridad para todos los dispositivos de la empresa. Sería de gran utilidad aportar una herramienta de este calibre a los administradores de sistemas de seguridad ya que, a día de hoy, no existe ninguna herramienta similar y facilitaría mucho el trabajo de los administradores, así como la seguridad corporativa.

Otra mejora a realizar sería el desarrollo o integración de interfaces visuales para las herramientas estudiadas. Esta mejora permitiría manejar las herramientas a usuarios con perfil más bajo. Por ejemplo, a la hora de monitorear la red mediante el sistema de detección de intrusos existen soluciones para Snort con las cuales poder visualizar los resultados de manera más ordenada. Una de ellas es BASE [22]. De forma análoga, se buscarían o implementarían soluciones similares para las herramientas restantes.

Sería muy interesante dar el salto a las demás plataformas móviles existentes a día de hoy: IOS, BlackBerry, etc. Con este desarrollo no se limitaría a ninguna empresa a usar dispositivos Android para poder gestionar sus políticas de seguridad aplicadas a los dispositivos móviles.

La seguridad no es un producto, sino que es un proceso sujeto a cambios que evoluciona constantemente a lo largo del tiempo. Es por ello, que otra de las líneas de mejora sería el desarrollo de un esquema integral de seguridad como base del MDM [23]. Este esquema incluiría la integración de herramientas de análisis forense en la fase de implantación, verificación y corrección de la política de seguridad [24, 25]. La integración de herramientas de este tipo pueden llegar a ser muy útiles debido al dilema BYOD. Con estas herramientas se podrían comprobar las trazas dejadas por documentos confidenciales en el dispositivo, por ejemplo.

Cabe destacar que, una vez finalizado este estudio, se continuará desarrollando e investigando, con el objetivo de desarrollar una aplicación que integre todas las herramientas estudiadas y poder comercializar, ya sea de forma gratuita o no, el resultado final del trabajo.

Capítulo 8

Conclusiones

Este documento ha consistido en la definición y el desarrollo de una política de seguridad para dispositivos Android en entornos empresariales. El proceso realizado ha consistido en estudiar el sistema operativo Android, analizar cómo definir una política de seguridad en un Sistema General de Información y desarrollar dicha política de seguridad (SGSI).

En lo referente al sistema operativo Android, se ha realizado una investigación a fondo del sistema operativo. Se ha analizado la arquitectura que posee, se ha estudiado como se gestionan los permisos y, por último, se han identificado las amenazas existentes en dichos dispositivos. A su vez, ha sido necesario aprender como realizar un *rooteo* a un dispositivo para lograr permisos de superusuario.

Mediante la definición de la política de seguridad se ha profundizado en los sistemas seguridad y en los sistemas generales de información. A su vez, se han estudiado principios, estándares y normativas de entidades importante en este sector como NIST. También se ha aprendido cómo clasificar la información en el ámbito empresarial en función de la importancia que ésta tenga. Por último, se ha estudiado cómo analizar y gestionar los riesgos existentes en una empresa.

La parte del documento dedicada al desarrollo de la política de seguridad ha sido en la que más esfuerzo se ha tenido que emplear al utilizar herramientas no vistas a lo largo de la carrera. A continuación, se exponen los aspectos técnicos más destacados a los cuales ha sido necesario enfrentarse:

- **SSH**. A la hora de establecer una comunicación segura entre el dispositivo Android y un ordenador ha sido necesario implementar un canal SSH.
- **VPN**. Con el fin de garantizar una navegación por Internet segura en los dispositivos Android, se ha implementado un servidor VPN por el que pasará todo el tráfico procedente de los dispositivos.
- **IPTables**. En el ámbito empresarial, un aspecto muy importante en las políticas de seguridad es el *firewall*. Para desarrollar un *firewall* ha sido necesario realizar un estudio del funcionamiento de IPTables para, posteriormente, realizar la implementación de reglas con las que filtrar el tráfico.
- **Kernel de Linux**. A la hora de modificar parámetros en el dispositivo Android, ha sido necesario comprender el funcionamiento del kernel de Linux que posee Android.

- **Snort**. Siguiendo con la política de seguridad, se ha implementado un sistema de detección de intrusos mediante Snort.
- **Mobile Device Management**. Como herramienta adicional y muy importante a la hora de gestionar los dispositivos de forma remota, se ha realizado un amplio estudio de los Mobile Device Management. A su vez, se ha utilizado uno de los existentes para la política de seguridad definida.

Por último, cabe destacar que este documento ha sido escrito en LaTex con el consecuente aprendizaje obligatorio que implica.

Bibliografía

[1] ISO, "ISO27001: Information Security Management System (ISMS) standard," *Online: http://www.27000.org/iso-27001.htm*, Oct. 2005.

[2] G. Stoneburner, C. Hayden, and A. Feringa, "Nist special publication 800-27 rev a," 2004.

[3] W. Stallings, *Network Security Essentials - Applications and Standards (4. ed., internat. ed.)*. Pearson Education, 2010, 0136108059.

[4] OWASP. Top 10 mobile risks. Accessed July-2014. [Online]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks

[5] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, and S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms," Dec. 2009. [Online]. Available: http://arxiv.org/abs/0912.5101

[6] Google, "Android manifest." [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[7] P. O. Rai, *Android Application Security Essentials*. Packt Publishing, 2013, 1849515603, 9781849515603.

[8] Google, "Binder." [Online]. Available: http://developer.android.com/reference/android/os/Binder.html

[9] Wikipedia, "Firma digital." [Online]. Available: http://es.wikipedia.org/wiki/Firma_digital

[10] Bring your own device. Accessed July-2014. [Online]. Available: http://spectrum.ieee.org/computing/it/the-bring-your-own-device-dilemma

[11] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of OSDI 2010*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, Oct. 2010, pp. 1–6. [Online]. Available: http://appanalysis.org/tdroid10.pdf

[12] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in Android," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 239–252.

[13] A. Parrizas. Securely deploying Android devices. Accessed July-2014. [Online]. Available: http://www.sans.org/reading-room/whitepapers/sysadmin/securely-deploying-android-devices-33799

[14] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 5th ed. USA: Addison-Wesley Publishing Company, 2009, 0136079679, 9780136079675.

[15] M. Feilner and N. Graf, *Beginning OpenVPN 2.0.9*. Packt Publishing, 2009, 184719706X, 9781847197061.

[16] Iptables. [Online]. Available: http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall/doc-iptables-firewall.pdf

[17] Kaspersky, "Bluetooth." [Online]. Available: http://www.kaspersky.es/about/news/press/2013/Los_riesgos_de_la_conexion_Bluetooth_

[18] O. de la Seguridad de la Información, "Politica de password," Instituto Nacional de Tecnologías de la Comunicación, Tech. Rep., 2007. [Online]. Available: http://www.unirioja.es/servicios/si/seguridad/difusion/politica_contrasenas.pdf

[19] Google, "Device management google." [Online]. Available: http://developer.android.com/guide/topics/admin/device-admin.html

[20] Snort. Accessed July-2014. [Online]. Available: http://www.maestrosdelweb.com/editorial/snort/

[21] A. Parrizas. (2013) Monitoring network traffic for Android devices. Accessed July-2014. [Online]. Available: http://www.sans.org/reading-room/whitepapers/detection/monitoring-network-traffic-android-devices-34097

[22] Accessed July-2014. [Online]. Available: http://base.professionallyevil.com/

[23] J. Diaz, D. Arroyo, and F. Rodriguez, "A formal methodology for integral security design and verification of network protocols," *Journal of Systems and Software*, vol. 89, pp. 87–98, 2014.

[24] G. Grispos, W. B. Glisson, and T. Storer, "Using Smartphones as a Proxy for Forensic Evidence Contained in Cloud Storage Services," *2013 46th Hawaii International Conference on System Sciences*, pp. 4910–4919, Jan. 2013, 978-1-4673-5933-7. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6480436

[25] E. Oriwoh and P. Sant, "The Forensics Edge Management System: A Concept and Design," *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pp. 544–550, Dec. 2013, 978-1-4799-2482-0. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6726257

Glosario

API Interfaz de Programación de Aplicaciones. Es el conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción.

Apk Paquete generado a partir del código fuente de la aplicación.

BYOD Bring Your Own Device.

CYOD Choose Your Own Device.

Escalada de privilegios La escalada de privilegios se produce cuando un privilegio permite a un proceso realizar más tareas de las que debe hacer.

Framework Poner.

IDS Intrusion Detection Systems (Sistema de detección de Intrusos).

Kernel Es la parte fundamental del sistema operativo y es el que se encarga de dar acceso del hardware a los programas. También conocido como núcleo.

Manifiesto Fichero de una aplicación que almacena el conjunto de componentes de la aplicación y los módulos de seguridad.

MDM Mobile Device Management.

NFC Es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.

Principio de mínimo privilegio Principio según el cual los sujetos deben acceder exclusivamente a aquellos objetos que precisen inexcusablemente para ejecutar sus trabajos o procesos.

Rooteo Cambiar el estado de un dispositivo de usuario regular a usuario con privilegios y ventajas.

Sandbox Es el aislamiento de procesos. Es un mecanismo para ejecutar programas con seguridad y de forma separada.

Índice alfabético

Android, 9, 15–17, 27–37, 39–43, 45–47, 50, 52, 58–63, 71, 79, 81, 89

BCP, 20, 21, 86
BCP, 20
BYOD, 37–39, 79

ciberseguridad, 9
ciberterrorismo, 9

DRP, 20, 21, 86
DRP, 20

Foro Económico Mundial, 9

GID, 29, 33, 70

hacktivismo, 9

IDS, 10, 63–67
Intents, 31, 36

L2TP, 49
Linux, 27, 29, 33, 34, 40, 50, 52, 54, 55, 66, 67, 81

MDM, 16, 38, 39, 73–77
see Mobile Device Management, 10

PPTP, 48, 49

rooteo, 16, 45, 46, 81

SGSI, 24, 81, 87, 88
SSH, 46, 47, 59, 81

TIC, 16

UID, 29, 33

VPN, 47–52, 56, 57, 61, 71, 81, 90

Apéndice A

Cláusulas Normativa ISO 27001

Cláusula 4

La organización debe establecer, implementar, desplegar, supervisar, revisar, mantener y mejorar un SGSI con un suficiente soporte documental y una coherencia respecto al conjunto de actividades de negocio y el riesgo que la empresa ha de afrontar.

- **Establecer y gestionar el SGSI**. La organización debe diferenciar en base a los siguientes aspectos.
  - Establecer el SGSI.
  - Implementar y desplegar el SGSI.
  - Supervisar y revisar el SGSI.
  - Mantener y mejorar el SGSI.
- **Requerimientos documentales**. El soporte documental se divide en:
  - General.
  - Control de documentos.
  - Control de registros.

Cláusula 5

Esta cláusula indica que debe haber un compromiso por parte de la dirección de la organización y una correcta gestión de recursos.

- **Compromiso de la dirección**. Debe existir un compromiso por parte de la dirección con el objetivo de que se asegure que los objetivos y los planes asociados al SGSI son establecidos. A su vez, se deben establecer roles y responsabilidades para la seguridad de la información, se debe comunicar al resto de la organización la importancia de la seguridad de la información y se deben proporcionar los recursos suficientes para implantar, desplegar, supervisar, revisar, mantener y mejorar el SGSI.
- **Gestión de recursos**. Debe haber una provisión adecuada de recursos. Por otro lado, la organización debe encargarse de entrenar y concienciar a todos sus empleados.

Cláusula 6

La cláusula 6 indica que se debe comprobar de forma periódica que los objetivos de control, los controles, procesos y procedimientos están en consonancia con el estándar y con las exigencias legales y reguladoras, que son coherentes con los requerimientos de seguridad del sistema, que son implementados y mantenidos de forma efectiva y que funciona según lo previsto.

Cláusula 7

El objetivo de esta cláusula es la revisión de la gestión de los SGSI.

Se debe revisar de forma general al menos una vez al año para garantizar la continuidad, la adecuación y la efectividad. Los resultados de las revisiones deben ser debidamente documentados y mantener registros al respecto.

Cláusula 8

Esta cláusula busca la mejora del SGSI. Para ello, aboga por una mejora continua basada en el resultado de las revisiones y auditorías. A su vez, esto se debe conseguir gracias a las acciones correctivas que deben evitar las no conformidades con el estándar. Por último, se deben mejorar las acciones preventivas eliminando las causas que pudieran generar discrepancias con el estándar.

Apéndice B

Certificados digitales en Android

Es muy importante destacar el papel de los certificados digitales. Un certificado digital es un fichero informático generado por una entidad de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa confirmando de esta manera su identidad digital en Internet. Dicho certificado es válido principalmente para autenticar a un usuario o a un sitio web en Internet, por lo que es necesaria la colaboración de un tercero que sea de confianza para cualquiera de las partes que participen en la comunicación. El nombre de esta entidad es Autoridad Certificadora, pudiendo ser un organismo público o una empresa reconocida en Internet.

Según la Sede Electrónica del Instituto Nacional de Estadística (España), un certificado electrónico sirve para:

- Autenticar la identidad del usuario, de forma electrónica, ante terceros.
- Firmar electrónicamente de forma que se garantice la integridad de los datos transmitidos y su procedencia.
- Cifrar datos para que sólo el destinatario del documento pueda acceder a su contenido.

Los certificados digitales son mecanismos de autenticación, los cuales son la base de las políticas de control de acceso y de distribución de privilegios. El certificado digital facilita el intercambio de información por vía telemática garantizando: autenticidad, confidencialidad, integridad y no repudio. (ver Fig.B.1)

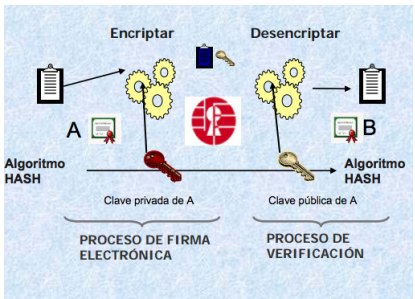

Figura B.1: Intercambio de información con el certificado digital

Android dispone de un repositorio (o almacén) de credenciales, también denominado

"almacenamiento de credenciales", protegido por mecanismos de cifrado, dónde se almacenan los certificados digitales, contraseñas y otras credenciales, como por ejemplo las asociadas a redes VPN y redes Wi-Fi.